C 8

```
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRR         RRR        III        VVV         VVV  EEE              RRR         RRR
DDD         DDD  RRRRRRRRRRRR           III        VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD         DDD  RRRRRRRRRRRR           III        VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD         DDD  RRRRRRRRRRRR           III        VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD         DDD  RRR   RRR              III        VVV         VVV  EEE              RRR   RRR
DDD         DDD  RRR   RRR              III        VVV         VVV  EEE              RRR   RRR
DDD         DDD  RRR   RRR              III        VVV         VVV  EEE              RRR   RRR
DDD         DDD  RRR      RRR           III           VVV   VVV     EEE              RRR      RRR
DDD         DDD  RRR      RRR           III           VVV   VVV     EEE              RRR      RRR
DDD         DDD  RRR      RRR           III           VVV   VVV     EEE              RRR      RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII           VVV        EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII           VVV        EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII           VVV        EEEEEEEEEEEEEEE  RRR         RRR
```

K 16

```
**FILE**ID**LADRIVER
```

```
LL            AAAAAA    DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
LL            AAAAAA    DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
LL          AA      AA  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
LL          AA      AA  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
LL          AA      AA  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
LL          AA      AA  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
LL          AA      AA  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
LL          AA      AA  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
LL          AAAAAAAAAA  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
LL          AAAAAAAAAA  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
LL          AA      AA  DD      DD  RR    RR        II        VV  VV    EE          RR    RR          ....
LL          AA      AA  DD      DD  RR    RR        II        VV  VV    EE          RR    RR          ....
LLLLLLLLLL  AA      AA  DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....
LLLLLLLLLL  AA      AA  DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0000          .TITLE  LADRIVER - LPA-11 DRIVER
0000          .IDENT  'V04-000'
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000
0000    ;++
0000    ; FACILITY:     EXECUTIVE, I/O DRIVERS
0000    ;
0000    ; ABSTRACT:
0000    ;       THIS MODULE IS THE DRIVER FOR THE LPA-11 (LABORATORY PERIPHERAL
0000    ;       ACCELERATOR).
0000    ;
0000    ; ENVIRONMENT:  KERNEL MODE, NON-PAGED
0000    ;
0000    ; AUTHOR:  STEVE BECKHARDT,     CREATION DATE:  7-APR-78
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-004 RNH0001         Richard N. Holstein     28-Aug-1984
0000    ;               Missing number sign in V03-002 caused ACCVIO.
0000    ;
0000    ;       V03-003 KDM0059         Kathleen D. Morse       14-Jul-1983
0000    ;               Change time-wait loop to use new TIMEDWAIT macro.
0000    ;               Add $DEVDEF.
0000    ;
0000    ;       V03-002 LJA0072         Laurie J. Anderson      17-Jun-1983
0000    ;               Correct DODIAGERL to properly recover from insufficient space
0000    ;               in error log buffers error condition.
0000    ;
0000    ;       V03-001 KDM0002         Kathleen D. Morse       28-Jun-1982
0000    ;               Added $DCDEF and $SSDEF.
0000    ;
0000    ;--
```

```
                   0000          .SBTTL  DECLARATIONS
                   0000  ;
                   0000  ; INCLUDE FILES:
                   0000  ;
                   0000          $ACBDEF                                 ; AST CONTROL BLOCK OFFSETS
                   0000          $ADPDEF                                 ; ADP OFFSETS
                   0000          $CCBDEF                                 ; CCB OFFSETS
                   0000          $CRBDEF                                 ; CRB OFFSETS
                   0000          $DCDEF                                  ; DEFINE DEVICE TYPE CODES
                   0000          $DDBDEF                                 ; DDB OFFSETS
                   0000          $DEVDEF                                 ; DEFINE DEVICE CHARACTERISTICS
                   0000          $DPTDEF                                 ; DRIVER PROLOGUE TABLE DEFINITIONS
                   0000          $DYNDEF                                 ; STRUCTURE TYPE CODE DEFINITIONS
                   0000          $EMBDEF                                 ; EMB OFFSETS
                   0000          $FKBDEF                                 ; FKB OFFSETS
                   0000          $IDBDEF                                 ; IDB OFFSETS
                   0000          $IPLDEF                                 ; IPL DEFINITIONS
                   0000          $IODEF                                  ; I/O FUNCTION CODES
                   0000          $IRPDEF                                 ; IRP OFFSETS
                   0000          $LADEF                                  ; LPA-11 DEFINITIONS
                   0000          $PCBDEF                                 ; PCB OFFSETS
                   0000          $PRDEF                                  ; PROCESSOR REGISTER DEFINITIONS
                   0000          $PRIDEF                                 ; PRIORITY INCREMENT CLASS DEFINITIONS
                   0000          $SSDEF                                  ; SYSTEM STATUS CODES
                   0000          $UCBDEF                                 ; UCB OFFSETS
                   0000          $VADEF                                  ; VIRTUAL ADDRESS FIELD DEFINITIONS
                   0000          $VECDEF                                 ; INTERRUPT DISPATCH VECTOR OFFSETS
                   0000
                   0000  ;
                   0000  ; MACROS:
                   0000  ;
                   0000
                   0000  ;
                   0000  ; EQUATED SYMBOLS:
                   0000  ;
                   0000
                   0000  ;
                   0000  ; QIO ARGUMENT LIST OFFSETS
                   0000  ;
                   0000
         00000000  0000  P1=0
         00000004  0000  P2=4
         00000008  0000  P3=8
         0000000C  0000  P4=12
                   0000
                   0000  ;
                   0000  ; MISC. DEFINITIONS
                   0000  ;
                   0000
         00000002  0000  DEVADDR=2                               ; OFFSET TO DEVICE ADDRESSES IN DMDT
         00000003  0000  STOP_MODE=3                             ; MODE FOR STOP RDA
         00000048  0000  IRP$L_SIP=IRP$L_SEGVBN                  ; POINTER TO SIP IN IRP
         0000003C  0000  IRP$L_BFR_AST=IRP$B_CARCON              ; BUFFER FULL AST ADDRESS IN IRP
         00000040  0000  IRP$L_OVR_AST=IRP$W_ABCNT               ; BUFFER OVERRUN AST ADDRESS IN IRP
         00000040  0000  IRP$L_RDAMAPREG=IRP$W_ABCNT             ; MAP REG. ALLOCATED FOR INTIALIZE
                   0000
                   0000  ;
```

```
          0000  ; LPA-11 DEVICE REGISTER OFFSETS
          0000  ;
          0000
          0000          $DEFINI LA
          0000
          0000  $DEF    LA_CISR         .BLKW   1       ; CONTROL IN STATUS REGISTER
          0002          _VIELD  LA_CISR,0,<-
          0002                  <GO,,M>,-               ; GO BIT
          0002                  <,1>,-                  ; RESERVED BIT
          0002                  <MEX,2>,-               ; MEMORY EXTENSIOON BITS
          0002                  <,2>,-                  ; RESERVED BITS
          0002                  <IE,,M>,-               ; READY IN INTERRUPT ENABLE
          0002                  <RDY,,M>,-              ; READY IN
          0002                  <,2>,-                  ; RESERVED BITS
          0002                  <ROMO,,M>,-             ; ROM OUTPUT BIT
          0002                  <ENA,,M>,-              ; ENABLE ARBITRATION
          0002                  <,1>,-                  ; RESERVED BIT
          0002                  <CRAM,,M>,-             ; CRAM WRITE
          0002                  <RESET,,M>,-            ; RESET (MASTER CLEAR)
          0002                  <RUN,,M>,-              ; RUN
          0002          >
          0002
          0002  $DEF    LA_COSR         .BLKW   1       ; CONTROL OUT STATUS REGISTER
          0004          _VIELD  LA_COSR,0,<-
          0004                  <USER,3>,-              ; USER INDEX
          0004                  <,3>,-                  ; RESERVED BITS
          0004                  <IE,,M>,-               ; READY OUT INTERRUPT ENABLE
          0004                  <RDY,,M>,-              ; READY OUT
          0004                  <ERRCD,5>,-             ; ERROR CODE
          0004                  <ERRTP,2>,-             ; ERROR TYPE
          0004                  <ERROR,,M>,-            ; ERROR BIT
          0004          >
          0004
          0004  $DEF    LA_RDA          .BLKW   1       ; RDA ADDRESS REGISTER
          0006
          0006  $DEF    LA_MAINT        .BLKW   1       ; MAINTENANCE STATUS REGISTER
          0008
          0008          $DEFEND LA
          0000
          0000
          0000  ;
          0000  ; LPA-11 SPECIFIC UCB OFFSETS
          0000  ;
          0000
          0000          $DEFINI UCB
          0000
000000A0  0000  .=UCB$L_DPC+4
          00A0
          00A0  $DEF    UCB$L_RDABA     .BLKL   1       ; UNIBUS ADDRESS OF RDA IN UCB
          00A4  $DEF    UCB$L_RDAMR     .BLKL   1       ; RDA IN UCB MAP REGISTER INFO.
          00A8  $DEF    UCB$L_PREALLOC  .BLKL   1       ; PREALLOCATED MAP REGISTER INFO.
          00AC  $DEF    UCB$L_INQFL     .BLKL   1       ; INPUT QUEUE FORWARD LINK
          00B0  $DEF    UCB$L_INQBL     .BLKL   1       ; INPUT QUEUE BACKWARD LINK
          00B4  $DEF    UCB$L_FORKO     .BLKL   6       ; READY OUT INTERRUPTS FORK BLOCK
          00CC  $DEF    UCB$L_FORKP     .BLKL   6       ; POWER RECOVERY FORK BLOCK
          00E4  $DEF    UCB$L_REGSAVE   .BLKL   4       ; REGISTER SAVE AREA
          00F4  $DEF    UCB$W_RISAVE    .BLKW   4       ; REG. SAVE AREA FOR READY-IN INTERRUPTS
```

```
                    00FC   172 $DEF     UCB$W_ROSAVE     .BLKW    4       ; REG. SAVE AREA FOR READY-OUT INTS.
                    0104   173 $DEF     UCB$L_RQLIST     .BLKL    8       ; USER REQUEST LIST
                    0124   174 $DEF     UCB$W_MRBITMAP   .BLKW    31      ; MAP REGISTER BITMAP
        00000164    0162   175                           .BLKW    1       ; SPARE WORD
                    0164   176 $DEF     UCB$W_RDA        .BLKW    29      ; RDA
        000001A0    019E   177                           .BLKW    1       ; SPARE WORD
                    01A0   178
        000001A0    01A0   179 UCB$K_SIZE=.
                    01A0   180
                    01A0   181          $DEFEND UCB
                    0000   182
                    0000   183 ;
                    0000   184 ;        SECONDARY I/O PACKET (SIP) OFFSETS
                    0000   185 ;
                    0000   186          $DEFINI SIP
                    0000   187
                    0000   188 $DEF     SIP$W_MODE       .BLKW    1       ; LPA-11 MODE WORD
                    0002   189 $DEF     SIP$W_BCNT       .BLKW    1       ; SIZE OF EACH BUFFER (IN BYTES)
        00000007    0004   190                           .BLKB    3       ; SPARE BYTES
                    0007   191 $DEF     SIP$B_VBFRMASK   .BLKB    1       ; VALID BUFFER MASK
                    0008   192 $DEF     SIP$W_SIZE       .BLKW    1       ; SIZE OF SIP
                    000A   193 $DEF     SIP$B_TYPE       .BLKB    1       ; TYPE OF DATA STRUCTURE
        0000000C    000B   194                           .BLKB    1       ; SPARE
                    000C   195 $DEF     SIP$L_SLVDATA    .BLKL    4       ; SLAVE DATA
                    001C   196 $DEF     SIP$L_USW_SVAPT  .BLKL    1       ; USW SVAPTE
                    0020   197 $DEF     SIP$W_USW_BOFF   .BLKW    1       ; USW BYTE OFFSET
                    0022   198 $DEF     SIP$W_USW_BCNT   .BLKW    1       ; USW BYTE COUNT
                    0024   199 $DEF     SIP$W_USW_MAPRE  .BLKW    1       ; USW STARTING MAP REGISTER
                    0026   200 $DEF     SIP$B_USW_NUMRE  .BLKB    1       ; USW NUMBER OF MAP REGISTERS
                    0027   201 $DEF     SIP$B_USW_DATAP  .BLKB    1       ; USW DATAPATH #
                    0028   202 $DEF     SIP$L_BFR_SVAPT  .BLKL    1       ; BFR SVAPTE
                    002C   203 $DEF     SIP$W_BFR_BOFF   .BLKW    1       ; BFR BYTE OFFSET
                    002E   204 $DEF     SIP$W_BFR_BCNT   .BLKW    1       ; BFR BYTE COUNT
                    0030   205 $DEF     SIP$W_BFR_MAPRE  .BLKW    1       ; BFR STARTING MAP REGISTER
                    0032   206 $DEF     SIP$B_BFR_NUMRE  .BLKB    1       ; BFR NUMBER OF MAP REGISTERS
                    0033   207 $DEF     SIP$B_BFR_DATAP  .BLKB    1       ; BFR DATAPATH #
                    0034   208 $DEF     SIP$L_RCL_SVAPT  .BLKL    1       ; RCL SVAPTE
                    0038   209 $DEF     SIP$W_RCL_BOFF   .BLKW    1       ; RCL BYTE OFFSET
                    003A   210 $DEF     SIP$W_RCL_BCNT   .BLKW    1       ; RCL BYTE COUNT
                    003C   211 $DEF     SIP$W_RCL_MAPRE  .BLKW    1       ; RCL STARTING MAP REGISTER
                    003E   212 $DEF     SIP$B_RCL_NUMRE  .BLKB    1       ; RCL NUMBER OF MAP REGISTERS
                    003F   213 $DEF     SIP$B_RCL_DATAP  .BLKB    1       ; RCL DATAPATH #
                    0040   214
                    0040   215          $DEFEND SIP
                    0000   216
```

```
0000  ;
0000  ; OWN STORAGE:
0000  ;
0000
0000  ;
0000  ; DRIVER PROLOGUE TABLE
0000  ;
0000          DPTAB   END=LA_END,-                ; END OF DRIVER
0000                  ADAPTER=UBA,-               ; ADAPTER TYPE
0000                  FLAGS=DPT$M_NOUNLOAD,-      ; DRIVER IS NOT RELOADABLE
0000                  UCBSIZE=UCB$K_SIZE,-        ; UCB SIZE
0000                  NAME=LADRIVER               ; DRIVER NAME
0038
0038          DPT_STORE INIT
0038          DPT_STORE UCB,UCB$B_FIPL,B,8        ; FORK IPL
003C          DPT_STORE UCB,UCB$L_DEVCHAR,L,-     ; DEVICE CHARACTERISTICS
003C                  <DEV$M_RTM-                 ; REAL TIME DEVICE
003C                  !DEV$M_AVL-                 ; AVAILABLE
003C                  !DEV$M_SHR-                 ; SHAREABLE
003C                  !DEV$M_ELG-                 ; ERROR LOGGING ENABLED
003C                  !DEV$M_IDV-                 ; INPUT DEVICE
003C                  !DEV$M_ODV>                 ; OUTPUT DEVICE
0043          DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_REALTIME         ; DEVICE CLASS
0047          DPT_STORE UCB,UCB$B_DEVTYPE,B,DT$_LPA11             ; DEVICE TYPE
004B          DPT_STORE UCB,UCB$B_DIPL,B,22                       ; DEVICE IPL
004F          DPT_STORE UCB,UCB$L_FORKO+8,L,-                     ; READY OUT FORK BLOCK
004F                  <<8@24>+<DYN$C_FRK@16>+FKB$K_LENGTH>        ; SIZE, TYPE, AND IPL
0056          DPT_STORE UCB,UCB$L_FORKP+8,L,-                     ; POWER REC. FORK BLOCK
0056                  <<8@24>+<DYN$C_FRK@16>+FKB$K_LENGTH>        ; SIZE, TYPE, AND IPL
005D
005D          DPT_STORE REINIT
005D          DPT_STORE DDB,DDB$L_DDT,D,LA$DDT                    ; DDT ADDRESS
0062          DPT_STORE CRB,CRB$L_INTD+4,D,LA$RDYOUTINTSV         ; READY OUT INT. SERVICE
0067          DPT_STORE CRB,CRB$L_INTD2+4,D,LA$RDYININTSV         ; READY IN INT. SERVICE
006C          DPT_STORE CRB,CRB$L_INTD+VEC$L_UNITINIT,D,UNIT_INIT ; UNIT INIT
0071          DPT_STORE END
0000
0000  ;
0000          DRIVER DISPATCH TABLE
0000  ;
0000          DDTAB   LA,-                        ; DEVICE NAME
0000                  STARTIO,-                   ; START I/O ENTRY POINT
0000                  0,-                         ; UNSOLICITED INTERRUPT
0000                  FUNCTABLE,-                 ; FUNCTION DECISION TABLE
0000                  CANCEL_IO,-                 ; CANCEL I/O
0000                  LA_REGDUMP,-                ; REGISTER DUMP ROUTINE
0000                  <36+24>,-                   ; SIZE OF DIAGNOSTIC BUFFER
0000                  <EMB$L_DV_REGSAV+4+24>      ; SIZE OF ERROR LOGGING BUFFER
0038
0038
0038  ;
0038          FUNCTION DECISION TABLE
0038  ;
0038  FUNCTABLE:
0038          FUNCTAB ,<LOADMCODE,STARTMPROC,-            ; LEGAL FUNCTIONS
0038                  INITIALIZE,SETCLOCK,SETCLOCKP,-
0038                  STARTDATA,STARTDATAP,-
```

```
              0038                   QSTOP>
              0040          FUNCTAB                               ; NO BUFFERED I/O FUNCTIONS
              0048          FUNCTAB LOAD_MICROCODE,<LOADMCODE>    ; LOAD MICROCODE
              0054          FUNCTAB STARTMP_FDT,<STARTMPROC>      ; START MICROPROCESSOR
              0060          FUNCTAB INIT_FDT,<INITIALIZE>         ; INITIALIZE
              006C          FUNCTAB SETCLOCK_FDT,<SETCLOCK,-      ; SET CLOCK
              006C                            SETCLOCKP>          ; SET CLOCK (PHYSICAL)
              0078          FUNCTAB STARTDATA_FDT,<STARTDATA,-    ; START DATA
              0078                            STARTDATAP>         ; START DATA (PHYSICAL)
              0084          FUNCTAB QSTOP_FDT,<QSTOP>             ; QUEUE STOP
              0090
              0090
              0090   ;
              0090   ; THE FOLLOWING TABLE IS USED FOR DISPATCHING IN STARTIO.
              0090   ; THE ORDER OF THE ENTRIES MUST NOT BE CHANGED!
              0090   ;
              0090   IOFCTBL:  ; I/O FUNCTION CODE TABLE - USED FOR DISPATCHING IN STARTIO
        02    0090          .BYTE   IO$_STARTMPROC
        04    0091          .BYTE   IO$_INITIALIZE
        37    0092          .BYTE   IO$_SETCLOCK
        05    0093          .BYTE   IO$_SETCLOCKP
        38    0094          .BYTE   IO$_STARTDATA
        06    0095          .BYTE   IO$_STARTDATAP
        03    0096          .BYTE   IO$_STOP
  00000007    0097   IOFCTBLN=.-IOFCTBL
```

```
                        0097          .SBTTL  LOAD_MICROCODE - FDT ROUTINE TO LOAD MICROCODE
                        0097
                        0097  ;++
                        0097  ; FUNCTIONAL DESCRIPTION:
                        0097  ;
                        0097  ;       THIS ROUTINE IS AN FDT ROUTINE WHICH PERFORMS THE LOAD MICROCODE
                        0097  ;       QIO.  IT LOCKS THE MICROCODE IMAGE IN MEMORY, CHECKS FOR NO ONGOING
                        0097  ;       DATA TRANSFERS, MASTER CLEAR'S THE LPA-11, CLEARS THE MICROCODE VALID
                        0097  ;       BIT, AND LOADS AND VERIFIES THE MICROCODE.  AFTER A SUCCESSFUL LOAD,
                        0097  ;       THE SHAREABLE BIT IS SET IF MULTIREQUEST MODE MICROCODE WAS LOADED
                        0097  ;       AND CLEARED OTHERWISE.  ALSO, THE MICROCODE TYPE IS SAVED AND THE
                        0097  ;       MICROCODE VALID BIT IS SET.
                        0097  ;
                        0097  ; CALLING SEQUENCE:
                        0097  ;
                        0097  ;       CALLED FROM THE FDT ROUTINE DISPATCHER IN THE QIO SYSTEM SERVICE.
                        0097  ;       ON COMPLETION JUMPS TO EXE$FINISHIOC.
                        0097  ;
                        0097  ; INPUT PARAMETERS:
                        0097  ;
                        0097  ;       R3      ADDRESS OF I/O PACKET
                        0097  ;       R4      CURRENT PROCESS PCB ADDRESS
                        0097  ;       R5      ADDRESS OF UCB
                        0097  ;       R6      ADDRESS OF CCB
                        0097  ;       AP      ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER
                        0097  ;
                        0097  ; OUTPUT PARAMETERS:
                        0097  ;
                        0097  ;       R0      THE LOW ORDER WORD CONTAINS A COMPLETION CODE;
                        0097  ;               THE HIGH ORDER WORD CONTAINS THE NUMBER OF BYTES OF
                        0097  ;               MICROCODE LOADED.
                        0097  ;
                        0097  ; COMPLETION CODES:
                        0097  ;
                        0097  ;       THESE ARE IN ADDITION TO THE ONES EXE$WRITELOCK CAN RETURN:
                        0097  ;
                        0097  ;       SS$_NORMAL      NORMAL
                        0097  ;       SS$_DATACHECK   MICROCODE LOAD ERROR
                        0097  ;       SS$_DEVACTIVE   DEVICE ACTIVE
                        0097  ;
                        0097  ; SIDE EFFECTS:
                        0097  ;
                        0097  ;       R1,R2,R4,R9,R10 ARE NOT SAVED
                        0097  ;
                        0097  ;--
                        0097
                        0097  LOAD_MICROCODE:
       50    6C    D0   0097          MOVL    P1(AP),R0               ; ADDRESS OF MICROCODE IMAGE
   51    04 AC     3C   009A          MOVZWL  P2(AP),R1               ; LENGTH OF IMAGE
       59    50    7D   009E          MOVQ    R0,R9                   ; PUT ADDRESS, SIZE INTO R9, R10
 00000000'GF       16   00A1          JSB     G^EXE$WRITELOCK         ; LOCK IT DOWN
    64 A5    20    AA   00A7          BICW    #UCB$M_POWER,UCB$W_STS(R5)  ; CLEAR POWERFAIL BIT
                        00AB
                        00AB  5$:     ; COME HERE TO TRY AGAIN AFTER A POWERFAIL
       50    59    7D   00AB          MOVQ    R9,R0                   ; RESTORE R0, R1
                        00AE
                        00AE          ; RESET MICROPROCESSOR
```

```
                       00AE          DSBINT  UCB$B_FIPL(R5)            ; RAISE IPL TO FORK LEVEL
          0078    30   00B5          BSBW    RESET
            01    CA   00B8          BICL    #LA$M_MCVALID,-           ; CLEAR MICROCODE VALID BIT
         44 A5         00BA                  UCB$L_DEVDEPEND(R5)
                       00BC          ENBINT                            ; LOWER IPL
     52   08 AC   3C   00BF          MOVZWL  P3(AP),R2                 ; GET MICRO PC TO START LOADING AT
            00    DD   00C3          PUSHL   #0                        ; COUNTER OF WORDS LOADED
 51  51   FF 8F   78   00C5          ASHL    #-1,R1,R1                 ; CONVERT BYTE TO WORD COUNT
            32    13   00CA          BEQL    15$                       ; WORD COUNT = 0
                       00CC
                       00CC   10$:   ; LOAD NEXT MICROCODE WORD
            64    B4   00CC          CLRW    LA_CISR(R4)               ; CLEAR CONTROL IN STATUS REGISTER
      04 A4 52    B0   00CE          MOVW    R2,LA_RDA(R4)             ; ADDRESS TO LOAD
      06 A4 60    B0   00D2          MOVW    (R0),LA_MAINT(R4)         ; MICROCODE WORD BEING LOADED
  64   0400 8F    B0   00D6          MOVW    #LA_CISR_M_ROMO,LA_CISR(R4)  ; SELECT ADDRESS
  64   2000 8F    A8   00DB          BISW    #LA_CISR_M_CRAM,LA_CISR(R4)  ; SET CRAM WRITE
            64    B4   00E0          CLRW    LA_CISR(R4)               ; RESET
                       00E2
                       00E2          ; NOW VERIFY WORD WAS LOADED CORRECTLY
      04 A4 52    B0   00E2          MOVW    R2,LA_RDA(R4)             ; MICRO ADDRESS
  64   0400 8F    B0   00E6          MOVW    #LA_CISR_M_ROMO,LA_CISR(R4)  ; SELECT CRAM AT ADDRESS
      06 A4 80    B1   00EB          CMPW    (R0)+,LA_MAINT(R4)        ; COMPARE CONTENTS WITH ORIGINAL WORD
            12    12   00EF          BNEQ    20$                       ; ERROR - NOT EQUAL
            52    B6   00F1          INCW    R2                        ; ADD 1 TO MICRO PC
      D5 6E 51    F2   00F3          AOBLSS  R1,(SP),10$               ; GO BACK AND LOAD NEXT WORD
                       00F7
                       00F7          ; SUCCESSFUL LOAD
 02  01   FE A0   F0   00F7          INSV    -2(R0),#LA$V_MCTYPE,#LA$S_MCTYPE,-  ; STORE MICROCODE TYPE
         44 A5         00FC                  UCB$L_DEVDEPEND(R5)       ; IN DEVICE DEPENDENT CHARACTERISTICS
      50    01    3C   00FE   15$:   MOVZWL  S^#SS$_NORMAL,R0
            05    11   0101          BRB     30$
                       0103
                       0103   20$:   ; ERROR DURING LOAD
    50   005C 8F  3C   0103          MOVZWL  #SS$_DATACHECK,R0
                       0108
                       0108   30$:   ; CONVERT # OF WORDS LOADED TO BYTES AND STORE IN HIGH WORD OF R0
 50  0F   11 8E   F0   0108          INSV    (SP)+,#17,#15,R0
                       010D          ; IF POWERFAIL OCCURRED THEN RETRY
                       010D          DSBINT  #31
   06 64 A5 05    E5   0113          BBCC    #UCB$V_POWER,UCB$W_STS(R5),40$  ;  BRANCH IF POWER DIDN'T FAIL
                       0118          ENBINT                            ; POWERFAIL OCCURRED, RETRY
          FF8D    31   011B          BRW     5$
                       011E
                       011E   40$:   ; NO POWERFAIL - IF SUCCESSFUL LOAD, THEN SET MICROCODE VALID
      50    01    B1   011E          CMPW    S^#SS$_NORMAL,R0          ; SUCCESSFUL?
            04    12   0121          BNEQ    50$                       ; NO
            01    88   0123          BISB    #LA$M_MCVALID,-           ; YES, SET MICROCODE VALID BIT
         44 A5         0125                  UCB$L_DEVDEPEND(R5)
                       0127   50$:   ENBINT
   00000000'GF    17   012A          JMP     G^EXE$FINISHIOC           ; RETURN TO USER
```

```
                          0130          .SBTTL  RESET - RESET MICROPROCESSOR
                          0130
                          0130    ;++
                          0130    ; FUNCTIONAL DESCRIPTION:
                          0130    ;
                          0130    ;       THIS ROUTINE VERIFIES THAT THERE ARE NO ONGOING DATA TRANSFERS,
                          0130    ;       AND THAT THE UCB IS NOT BUSY. IF THESE CONDITIONS ARE MET, THEN
                          0130    ;       A MASTER CLEAR IS ISSUED TO THE LPA-11.  OTHERWISE, THE I/O
                          0130    ;       IS FINISHED WITH AN ERROR STATUS.  THIS ROUTINE MUST BE CALLED
                          0130    ;       AT FORK IPL TO AVOID RACE CONDITIONS.
                          0130    ;
                          0130    ; CALLING SEQUENCE:
                          0130    ;
                          0130    ;       BSBW    RESET
                          0130    ;
                          0130    ; INPUT PARAMETERS:
                          0130    ;
                          0130    ;       R5      ADDRESS OF UCB
                          0130    ;
                          0130    ; IMPLICIT INPUTS:
                          0130    ;
                          0130    ;       IPL IS AT FORK LEVEL ON ENTRY
                          0130    ;
                          0130    ; OUTPUT PARAMETERS:
                          0130    ;
                          0130    ;       R4      UNIBUS ADDRESS OF FIRST LPA-11 REGISTER
                          0130    ;
                          0130    ; COMPLETION CODES:
                          0130    ;
                          0130    ;       SS$_DEVACTIVE   DEVICE ACTIVE (NOT RETURNED TO CALLER - GOES
                          0130    ;                       DIRECTLY TO EXE$FINISHIOC)
                          0130    ;
                          0130    ; SIDE EFFECTS:
                          0130    ;
                          0130    ;       R2 IS NOT PRESERVED
                          0130    ;--
                          0130
                          0130   RESET:
25 64 A5    08      E0    0130          BBS     #UCB$V_BSY,UCB$W_STS(R5),20$    ; MAKE SURE UCB IS NOT BUSY
                          0135
                          0135          ; MAKE SURE THERE ARE NO ONGOING DATA TRANSFERS
            52      D4    0135          CLRL    R2
   0104 C542        D5    0137   10$:   TSTL    UCB$L_RQLIST(R5)[R2]    ; A REQUEST HERE?
            1C      12    013C          BNEQ    20$                     ; YES, ERROR!
   F5 52    08      F2    013E          AOBLSS  #8,R2,10$               ; TRY NEXT SLOT
                          0142
                          0142          ; GET POINTER TO DEVICE REGISTERS
   54    24 A5      D0    0142          MOVL    UCB$L_CRB(R5),R4        ; GET POINTER TO CRB
                          0146          ASSUME  IDB$L_CSR EQ 0
   54    2C B4      D0    0146          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4  ; GET PTR TO 1ST DEVICE REGISTER
                          014A
                          014A          ; RAISE IPL TO HARDWARE DEVICE LEVEL AND DO A MASTER CLEAR
                          014A          DSBINT  UCB$B_DIPL(R5)
  64   4000 8F      B0    0151          MOVW    #LA_CISR_M_RESET,LA_CISR(R4)  ; DO MASTER CLEAR
                          0156          ENBINT
                    05    0159          RSB
                          015A
```

```
                        015A    466 20$:    ; ERROR - LPA-11 IS BUSY
   50   02C4 8F    3C   015A    467         MOVZWL  #SS$_DEVACTIVE,R0       ; STATUS
    00000000'GF    17   015F    468         JMP     G^EXE$FINISHIOC         ; FINISH I/O
```

```
                   0165          .SBTTL  STARTMP_FDT     START MICROPROCESSOR FDT ROUTINE
                   0165
                   0165  ;++
                   0165  ; FUNCTIONAL DESCRIPTION:
                   0165  ;
                   0165  ;       THIS ROUTINE IS THE FDT ROUTINE FOR THE START MICROPROCESSOR
                   0165  ;       QIO.  IT CHECKS FOR NO ACTIVE USERS, MASTER CLEARS THE LPA-11,
                   0165  ;       AND THEN QUEUES THE PACKET ONTO THE UCB'S INPUT QUEUE.
                   0165  ;
                   0165  ; CALLING SEQUENCE:
                   0165  ;
                   0165  ;       CALLED BY THE FDT ROUTINE DISPATCHER IN THE QIO SYSTEM SERVICE.
                   0165  ;       ON COMPLETION BRANCHES TO QUE_PKT
                   0165  ;
                   0165  ; INPUT PARAMETERS:
                   0165  ;
                   0165  ;       R3      ADDRESS OF I/O PACKET
                   0165  ;       R5      ADDRESS OF UCB
                   0165  ;
                   0165  ; OUTPUT PARAMETERS:
                   0165  ;
                   0165  ;       NONE
                   0165  ;
                   0165  ; COMPLETION CODES:
                   0165  ;
                   0165  ;       SS$_DEVACTIVE   DEVICE ACTIVE (GETS RETURNED DIRECTLY TO EXE$FINISHIOC)
                   0165  ;
                   0165  ; SIDE EFFECTS:
                   0165  ;
                   0165  ;       R2,R4 ARE NOT PRESERVED
                   0165  ;--
                   0165
                   0165  STARTMP_FDT:
                   0165          SETIPL  UCB$B_FIPL(R5)          ; RAISE IPL TO FORK LEVEL
       C5   10     0169          BSBB    RESET                   ; RESET MICROPROCESSOR
     01AC   31     016B          BRW     QUE_PKT                 ; INITIATE FUNCTION
```

```
                              016E           .SBTTL  INIT_FDT - INITIALIZE FDT ROUTINE
                              016E
                              016E   ;++
                              016E   ; FUNCTIONAL DESCRIPTION:
                              016E   ;
                              016E   ;       THIS ROUTINE IS THE FDT ROUTINE FOR THE INITIALIZE QIO.
                              016E   ;       IT CHECKS FOR SEVERAL ERRORS, LOCKS THE INITIALIZE TABLE INTO
                              016E   ;       MEMORY, AND FORMATS THE CONFIGURATION BITS WHICH GET STORED
                              016E   ;       IN THE DEVICE CHARACTERISTICS IF THE INITIALIZE IS SUCCESSFUL.
                              016E   ;
                              016E   ; CALLING SEQUENCE:
                              016E   ;
                              016E   ;       CALLED FROM THE FDT ROUTINE DISPATCHER IN THE QIO SYSTEM SERVICE.
                              016E   ;
                              016E   ; INPUT PARAMETERS:
                              016E   ;
                              016E   ;       R3      ADDRESS OF I/O PACKET
                              016E   ;       R4      CURRENT PROCESS PCB ADDRESS
                              016E   ;       R5      ADDRESS OF UCB
                              016E   ;       R6      ADDRESS OF CCB
                              016E   ;       AP      ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER
                              016E   ;
                              016E   ; OUTPUT PARAMETERS:
                              016E   ;
                              016E   ;       NONE
                              016E   ;
                              016E   ; COMPLETION CODES:
                              016E   ;
                              016E   ;       SS$_IVMODE      INVALID MODE
                              016E   ;       SS$_IVBUFLEN    INVALID BUFFER LENGTH
                              016E   ;       SS$_BUFNOTALIGN BUFFER NOT ALIGNED CORRECTLY
                              016E   ;       (THESE ERRORS GET RETURNED DIRECTLY TO EXE$FINISHIOC)
                              016E   ;--
                              016E
                              016E
                              016E   INIT_FDT:
      52   0324 8F     3C     016E           MOVZWL  #SS$_BUFNOTALIGN,R2     ; ASSUME ALIGNMENT ERROR
           50   6C     D0     0173           MOVL    P1(AP),R0               ; GET ADDRESS OF INITIALIZE TABLE
             3C 50     E8     0176           BLBS    R0,10$                  ; VERIFY IT'S WORD ALIGNED
           59   50     D0     0179           MOVL    R0,R9                   ; SAVE FOR LATER USE
      52   034C 8F     3C     017C           MOVZWL  #SS$_IVBUFLEN,R2        ; ASSUME INVALID LENGTH ERROR
        51   04 AC     3C     0181           MOVZWL  P2(AP),R1               ; GET LENGTH
  00000116 8F   51     D1     0185           CMPL    R1,#278                 ; IS IT THE RIGHT LENGTH?
                27     12     018C           BNEQ    10$                     ; NO - ERROR
     00000000'GF       16     018E           JSB     G^EXE$WRITELOCK         ; YES, LOCK IT DOWN
      52   0354 8F     3C     0194           MOVZWL  #SS$_IVMODE,R2          ; ASSUME INVALID MODE ERROR
           69   07     93     0199           BITB    #7,(R9)                 ; MAKE SURE MODE = INITIALIZE
                17     12     019C           BNEQ    10$                     ; IT DOESN'T - ERROR
                              019E
                              019E           ; BUILD CONFIGURATION BITS FOR DEVICE CHARACTERISTICS
                51     D4     019E           CLRL    R1                      ; LOOP COUNTER AND BIT POSITION
      52   02 A941     B0     01A0   5$:     MOVW    DEVADDR(R9)[R1],R2      ; GET DEVICE ADDRESS OF NEXT DEVICE
50    01   51   52     F0     01A5           INSV    R2,R1,#1,R0             ; STORE LOW BIT OF ADDRESS IN R0
        F2 51   0A     F2     01AA           AOBLSS  #10,R1,5$               ; DO NEXT DEVICE
        38 A3   50     D2     01AE           MCOML   R0,IRP$L_MEDIA(R3)      ; COMPLEMENT BITS AND SAVE
              0165     31     01B2           BRW     QUE_PKT                 ; QUEUE PACKET TO DRIVER
                              01B5
```

```
                                 01B5     565 10$:    ; ERROR - EITHER INCORRECT LENGTH, MODE NOT EQUAL TO INIT,
                                 01B5     566         ; OR NOT WORD ALIGNED.
                  50    52   D0  01B5     567         MOVL    R2,R0                       ; COMPLETION CODE
            00000000'GF     17   01B8     568         JMP     G^EXE$FINISHIOC
```

```
                              01BE          .SBTTL  SETCLOCK_FDT - SET CLOCK FDT ROUTINE
                              01BE
                              01BE  ;++
                              01BE  ; FUNCTIONAL DESCRIPTION:
                              01BE  ;
                              01BE  ;       THIS ROUTINE IS THE FDT ROUTINE FOR THE SET CLOCK QIO.
                              01BE  ;       IT COPIES THE FUNCTION DEPENDENT PARAMETERS INTO THE I/O
                              01BE  ;       PACKET AND THEN STORES THE CLOCK A RATE AND PRESET IN THE
                              01BE  ;       SPARE CHARACTERISTICS.  THIS WILL GET STORED IN THE DEVICE
                              01BE  ;       CHARACTERISTICS IF THE QIO IS SUCCESSFUL.
                              01BE  ;
                              01BE  ; CALLING SEQUENCE:
                              01BE  ;
                              01BE  ;       CALLED BY THE FDT ROUTINE DISPATCHER IN THE QIO SYSTEM SERVICE.
                              01BE  ;
                              01BE  ; INPUT PARAMETERS:
                              01BE  ;
                              01BE  ;       R3      ADDRESS OF I/O PACKET
                              01BE  ;       R5      UCB ADDRESS
                              01BE  ;       AP      ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER
                              01BE  ;
                              01BE  ; OUTPUT PARAMETERS:
                              01BE  ;
                              01BE  ;       NONE
                              01BE  ;--
                              01BE
                              01BE  SETCLOCK_FDT:
                              01BE          ; COPY P2 - P4 INTO I/O PACKET
        38 A3   04 AC    B0   01BE          MOVW    P2(AP),IRP$L_MEDIA(R3)    ; MODE WORD
        3A A3   08 AC    B0   01C3          MOVW    P3(AP),IRP$L_MEDIA+2(R3)  ; CLOCK STATUS
        3C A3   0C AC    B0   01C8          MOVW    P4(AP),IRP$L_MEDIA+4(R3)  ; CLOCK PRESET
  38 A3   03    00   01  F0   01CD          INSV    #1,#0,#3,IRP$L_MEDIA(R3)  ; SET MODE TO START CLOCK
                              01D3
                 0144    31   01D3          BRW     QUE_PKT                   ; QUEUE PACKET TO DRIVER
```

```
                          01D6             .SBTTL  STARTDATA_FDT - START DATA FDT ROUTINE
                          01D6
                          01D6   ;++
                          01D6   ; FUNCTIONAL DESCRIPTION:
                          01D6   ;
                          01D6   ;        THIS ROUTINE IS THE FDT ROUTINE FOR THE START DATA QIO.  IT
                          01D6   ;        ALLOCATES A SECONDARY I/O PACKET (SIP), LOCKS THE USW, BUFFERS,
                          01D6   ;        AND RCL INTO MEMORY AND LINKS THE SIP TO THE IRP.
                          01D6   ;
                          01D6   ; CALLING SEQUENCE:
                          01D6   ;
                          01D6   ;        CALLED FROM THE FDT ROUTINE DISPATCHER IN THE QIO SYSTEM SERVICE
                          01D6   ;
                          01D6   ; INPUT PARAMETERS:
                          01D6   ;
                          01D6   ;        R3      ADDRESS OF I/O PACKET
                          01D6   ;        R4      CURRENT PROCESS PCB ADDRESS
                          01D6   ;        R5      ADDRESS OF UCB
                          01D6   ;        R6      ADDRESS OF CCB
                          01D6   ;
                          01D6   ; OUTPUT PARAMETERS:
                          01D6   ;
                          01D6   ;        NONE
                          01D6   ;
                          01D6   ; COMPLETION CODES:
                          01D6   ;
                          01D6   ;        SS$_INSFMEM     INSUFFICIENT MEMEORY
                          01D6   ;        SS$_BUFNOTALIGN ALIGNMENT ERROR
                          01D6   ;        SS$_IVBUFLEN    INVALID BUFFER LENGTH
                          01D6   ;        (THESE ERRORS GET RETURNED DIRECTLY TO EXE$FINISHIOC)
                          01D6   ;
                          01D6   ; SIDE EFFECTS:
                          01D6   ;
                          01D6   ;        R1,R2,R7,R8 ARE NOT PRESERVED
                          01D6   ;--
                          01D6
                          01D6            .ENABL  LSB
                          01D6   STARTDATA_FDT:
                          01D6            ; FIRST CHECK THAT ARGUMENT BLOCK POINTED TO BY P1 IS THE CORRECT
                          01D6            ; LENGTH AND ACCESSIBLE
            5A      D4    01D6            CLRL    R10                     ; MEANS NO SIP IN CASE OF ERROR
 51    04   AC      3C    01D8            MOVZWL  P2(AP),R1               ; GET LENGTH
    28      51      D1    01DC            CMPL    R1,#40                  ; IS IT CORRECT LENGTH?
            03      13    01DF            BEQL    5$                      ; YES
          00E3      31    01E1            BRW     LENGTHERR               ; NO - ERROR
    50      6C      D0    01E4   5$:      MOVL    P1(AP),R0               ; YES, GET POINTER
     00000000'GF    16    01E7            JSB     G^EXE$WRITECHK          ; CHECK FOR READ ACCESS
    59      50      D0    01ED            MOVL    R0,R9                   ; R9 WILL STEP THRU ARGUMENT BLOCK
                          01F0
                          01F0            ; NOW ALLOCATE SECONDARY I/O PACKET (SIP)
 51    00C4 8F      3C    01F0            MOVZWL  #IRP$C_LENGTH,R1        ; LENGTH
            53      DD    01F5            PUSHL   R3                      ; SAVE R3
     00000000'GF    16    01F7            JSB     G^EXE$ALONONPAGED       ; ALLOCATE IT
    53      8E      D0    01FD            MOVL    (SP)+,R3                ; RESTORE R3
         08 50      E8    0200            BLBS    R0,10$                  ; SUCCESSFUL
 50    0124 8F      3C    0203            MOVZWL  #SS$_INSFMEM,R0         ; ERROR
          00C3      31    0208            BRW     ABORT
```

```
                              020B
                              020B  10$:   ; CLEAR PACKET AND PUT IN SIZE
                     3F   BB  020B         PUSHR   #^M<R0,R1,R2,R3,R4,R5>
 62   51   00   62   00   2C  020D         MOVC5   #0,(R2),#0,R1,(R2)           ; CLEAR PACKET
                     3F   BA  0213         POPR    #^M<R0,R1,R2,R3,R4,R5>
        08 A2  00C4  8F   B0  0215         MOVW    #IRP$C_LENGTH,IRP$W_SIZE(R2)
                5A   52   D0  021B         MOVL    R2,R10                       ; R10 WILL POINT TO SIP
                              021E
                              021E         ; START BUILDING SIP FROM ARGUMENT BLOCK
                6A   89   B0  021E         MOVW    (R9)+,SIP$W_MODE(R10)        ; COPY MODE WORD
      6A   03   00   02   F0  0221         INSV    #2,#0,#3,SIP$W_MODE(R10)     ; MAKE SURE FUNCTION = START DATA
                5B   89   3C  0226         MOVZWL  (R9)+,R11                    ; GET VALID BUFFER MASK
             07 AA   5B   90  0229         MOVB    R11,SIP$B_VBFRMASK(R10)      ; STORE IN SIP
           5B  FFF8  8F   AA  022D         BICW    #^XFFF8,R11                  ; MASK EVERYTHING BUT # OF BUFFERS
                     5B   D6  0232         INCL    R11                          ; ADD 1 TO GET TRUE # OF BUFFERS
                              0234
                              0234  20$:   ; CHECK AND LOCK USW
                50   89   D0  0234         MOVL    (R9)+,R0                     ; POINTER TO USW
                  59 50   E8  0237         BLBS    R0,45$                       ; BRANCH IF NOT WORD ALIGNED (ERROR)
                51   02   D0  023A         MOVL    #2,R1                        ; LENGTH OF USW
                   0094   30  023D         BSBW    READLOCK                     ; CHECK AND LOCK FOR WRITE ACCESS
        1C AA    2C A3    7D  0240         MOVQ    IRP$L_SVAPTE(R3),SIP$L_USW_SVAPT(R10) ; SAVE SVAPTE, BOFF, BCNT
                              0245
                              0245         ; CHECK DATA BUFFER AREA FOR PROPER ALIGNMENT AND SIZE RESTRICTIONS
                51   69   3C  0245         MOVZWL  (R9),R1                      ; LENGTH OF BUFFER AREA
                59   04   C0  0248         ADDL    #4,R9
                50   89   D0  024B         MOVL    (R9)+,R0                     ; POINTER TO BUFFER AREA
                50   03   D3  024E         BITL    #3,R0                        ; MAKE SURE ITS LONGWORD ALIGNED
                     6D   12  0251         BNEQ    ALIGNERR                     ; IT'S NOT - ERROR!
                     52   D4  0253         CLRL    R2
      58   52   51   5B   7B  0255         EDIV    R11,R1,R2,R8                 ; GET SIZE OF EACH DATA BUFFER
                     6B   13  025A         BEQL    LENGTHERR                    ; BUFFER LENGTH CAN'T BE ZERO!
                     58   D5  025C         TSTL    R8                           ; MAKE SURE REMAINDER IS ZERO
                     67   12  025E         BNEQ    LENGTHERR                    ; IT'S NOT - ERROR!
                  64 52   E8  0260         BLBS    R2,LENGTHERR                 ; BUFFER SIZE MUST BE A MULTIPLE
                              0263                                              ; OF 2 IN MULTIREQUEST MODE.
             05 6A   03   E0  0263         BBS     #3,SIP$W_MODE(R10),27$       ; BR. IF THIS IS A M.R. MODE REQUEST
                52   03   D3  0267         BITL    #3,R2                        ; BUFFER SIZE MUST BE A MULTIPLE
                              026A                                              ; OF 4 IN DEDICATED MODE.
                     5B   12  026A         BNEQ    LENGTHERR                    ; IT'S NOT - ERROR!
             02 AA   52   B0  026C  27$:   MOVW    R2,SIP$W_BCNT(R10)           ; STORE BUFFER SIZE IN SIP
                              0270
                              0270         ; NOW CHECK AND LOCK BUFFERS FOR READ OR WRITE ACCESS DEPENDING
                              0270         ; ON TRANSFER DIRECTION
                     6A   95  0270         TSTB    SIP$W_MODE(R10)              ; TEST FOR TRANSFER DIRECTION
                     04   19  0272         BLSS    30$
                     5E   10  0274         BSBB    READLOCK                     ; FROM LPA TO MEMORY
                     02   11  0276         BRB     40$
                     62   10  0278  30$:   BSBB    WRITELOCK                    ; FROM MEMORY TO LPA
        28 AA    2C A3    7D  027A  40$:   MOVQ    IRP$L_SVAPTE(R3),SIP$L_BFR_SVAPT(R10) ; SAVE SVAPTE, BOFF, BCNT
                              027F
                              027F         ; REPEAT FOR RCL
                51   69   3C  027F         MOVZWL  (R9),R1                      ; LENGTH OF RCL
                59   04   C0  0282         ADDL    #4,R9
                50   89   D0  0285         MOVL    (R9)+,R0                     ; ADDRESS OF RCL
           6A  0300  8F   B3  0288         BITW    #^X300,SIP$W_MODE(R10)       ; IS RCL SPECIFIED?
                     1A   12  028D         BNEQ    50$                          ; NO
                51     B5  028F            TSTW    R1                       ; YES, MAKE SURE LENGTH IS NOT ZERO
                34     13  0291            BEQL    LENGTHERR                ; IT IS ZERO - ERROR
             2A 50     E8  0293  45$:      BLBS    R0,ALIGNERR              ; RCL MUST BE WORD ALIGNED
             2E 51     E8  0296            BLBS    R1,LENGTHERR             ; AND A MULTIPLE OF 2 IN LENGTH
          57    50     7D  0299            MOVQ    R0,R7                    ; SAVE R0,R1 IN R7,R8
                3E     10  029C            BSBB    WRITELOCK                ; CHECK ACCESS AND LOCK DOWN
    34 AA    2C A3     7D  029E            MOVQ    IRP$L_SVAPTE(R3),SIP$L_RCL_SVAPT(R10) ; SAVE SVAPTE, BCNT, BOFF
  1E FF A748    07     E1  02A3            BBC     #7,-1(R7)[R8],LENGTHERR ; MAKE SURE END OF RCL HAS HIGH BIT SET
                           02A9
     0C AA      89     7D  02A9  50$:      MOVQ    (R9)+,SIP$L_SLVDATA(R10)  ; COPY SLAVE DATA
     14 AA      89     7D  02AD            MOVQ    (R9)+,SIP$L_SLVDATA+8(R10)
                           02B1            ASSUME  IRP$L_OVR_AST EQ IRP$L_BFR_AST+4
    3C A3    08 AC     7D  02B1            MOVQ    P3(AP),IRP$L_BFR_AST(R3)  ; COPY AST ADDRESSES
             2C A3     7C  02B6            CLRQ    IRP$L_SVAPTE(R3)         ; CLEAR SVAPTE, BCNT, AND BOFF IN IRP
       48 A3    5A     D0  02B9            MOVL    R10,IRP$L_SIP(R3)        ; LINK SIP TO IRP
              005A     31  02BD            BRW     QUE_PKT                  ; QUEUE PACKET TO DRIVER
                           02C0
                           02C0
                           02C0            ; ERRORS COME HERE
                           02C0
                           02C0  ALIGNERR:   ; ALIGNMENT ERROR
       50   0324 8F    3C  02C0            MOVZWL  #SS$_BUFNOTALIGN,R0
                05     11  02C5            BRB     60$
                           02C7
                           02C7  LENGTHERR:  ; INVALID LENGTH ERROR
       50   034C 8F    3C  02C7            MOVZWL  #SS$_IVBUFLEN,R0
                17     10  02CC  60$:      BSBB    CLEANUP                  ; UNLOCK PAGES, DEALLOCATE SIP
                           02CE
        00000000'GF    17  02CE  ABORT:    JMP     G^EXE$FINISHIOC
                           02D4
                           02D4
                           02D4            ; LOCAL SUBROUTINES
                           02D4
                           02D4  READLOCK:
        00000000'GF    16  02D4            JSB     G^EXE$READLOCKR          ; LOCK PAGES FOR WRITE ACCESS
                06     11  02DA            BRB     70$
                           02DC
                           02DC  WRITELOCK:
        00000000'GF    16  02DC            JSB     G^EXE$WRITELOCKR         ; LOCK PAGES FOR READ ACCESS
             0F 50     E8  02E2  70$:      BLBS    R0,90$                   ; BRANCH IF EVERYTHING IS OK
                           02E5
                           02E5            ; ERROR OR HAVE TO FAULT PAGES IN.  FALL THROUGH TO ...
                           02E5
                           02E5
                           02E5  CLEANUP:  ; UNLOCK PAGES AND DEALLOCATE SIP
                3F     BB  02E5            PUSHR   #^M<R0,R1,R2,R3,R4,R5>
             2C A3     7C  02E7            CLRQ    IRP$L_SVAPTE(R3)         ; CLEAR SVAPTE, BCNT, AND BOFF IN IRP
          55    5A     D0  02EA            MOVL    R10,R5                   ; ADDRESS OF SIP
                03     13  02ED            BEQL    80$                      ; NO SIP - NOTHING TO UNLOCK
              028A     30  02EF            BSBW    UNLOCK                   ; UNLOCK PAGES, DEALLOCATE SIP
                3F     BA  02F2  80$:      POPR    #^M<R0,R1,R2,R3,R4,R5>
                05     05  02F4  90$:      RSB                              ; RETURN TO CALLER OR COROUTINE
                           02F5            .DSABL  LSB
```

```
                        02F5          .SBTTL  QSTOP_FDT - QUEUE STOP FDT ROUTINE
                        02F5
                        02F5    ;++
                        02F5    ; FUNCTIONAL DESCRIPTION:
                        02F5    ;
                        02F5    ;       THIS ROUTINE IS AN FDT ROUTINE WHICH PERFORMS THE QUEUE STOP
                        02F5    ;       QIO.  NOTE THAT THIS QIO DOES NOT ITSELF STOP A DATA TRANSFER;
                        02F5    ;       RATHER IT QUEUES THE ORIGINAL START DATA I/O PACKET BACK TO THE
                        02F5    ;       DRIVER AS A STOP.  THEREFORE, THIS QIO COMPLETES AS SOON AS
                        02F5    ;       THE STOP IS QUEUED.  THE ORIGINAL START DATA COMPLETES AFTER THE
                        02F5    ;       DATA TRANSFER HAS ACTUALLY STOPPED.
                        02F5    ;
                        02F5    ; CALLING SEQUENCE:
                        02F5    ;
                        02F5    ;       CALLED FROM THE FDT ROUTINE DISPATCHER IN THE QIO SYSTEM SERVICE.
                        02F5    ;       ON COMPLETION JUMPS TO EXE$FINISHIOC.
                        02F5    ;
                        02F5    ; INPUT PARAMETERS:
                        02F5    ;
                        02F5    ;       R3      ADDRESS OF I/O PACKET
                        02F5    ;       R4      CURRENT PROCESS PCB ADDRESS
                        02F5    ;       R5      ADDRESS OF UCB
                        02F5    ;       AP      ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER
                        02F5    ;
                        02F5    ; OUTPUT PARAMETERS:
                        02F5    ;
                        02F5    ;       R0      COMPLETION CODE
                        02F5    ;
                        02F5    ; COMPLETION CODES:
                        02F5    ;
                        02F5    ;       SS$_NORMAL      NORMAL
                        02F5    ;       SS$_BADPARAM    NO SUCH REQUEST
                        02F5    ;
                        02F5    ; SIDE EFFECTS:
                        02F5    ;
                        02F5    ;       R2 IS NOT PRESERVED
                        02F5    ;--
                        02F5
                        02F5    QSTOP_FDT:
   52    04 AC    9A    02F5            MOVZBL  P2(AP),R2               ; GET REQUEST NUMBER
   52    F8 8F    8A    02F9            BICB    #^XF8,R2                ; CLEAR ALL BUT LOW THREE BITS
      50    14    3C    02FD            MOVZWL  #SS$_BADPARAM,R0        ; ASSUME ERROR
                        0300            SETIPL  UCB$B_FIPL(R5)          ; RAISE TO FORK IPL
     0104 C542    D5    0304            TSTL    UCB$L_RQLIST(R5)[R2]    ; IS THERE A REQUEST IN THIS SLOT?
            09    13    0309            BEQL    10$                     ; NO - ERROR
      50    2C    3C    030B            MOVZWL  #SS$_ABORT,R0           ; YES - QUEUE A STOP WITH ABORT STATUS
          04FF    30    030E            BSBW    QUEUE_STOP_REQ
      50    01    3C    0311            MOVZWL  S^#SS$_NORMAL,R0        ; RETURN NORMAL STATUS
    00000000'GF   17    0314    10$:    JMP     G^EXE$FINISHIOC         ; FINISH I/O
```

```
                                   031A          .SBTTL  QUE_PKT - QUEUE I/O PACKET TO DRIVER
                                   031A
                                   031A     ;++
                                   031A     ; FUNCTIONAL DESCRIPTION:
                                   031A     ;
                                   031A     ;       THIS ROUTINE IS JUMPED TO FROM AN FDT ROUTINE TO QUEUE AN
                                   031A     ;       I/O PACKET TO THE DRIVER.  IF THE DRIVER IS NOT BUSY, THEN
                                   031A     ;       THE DRIVER IS CALLED IMMEDIATELY.  THIS ROUTINE IS SIMILAR TO
                                   031A     ;       THE EXEC'S, EXCEPT IT USES A DIFFERENT QUEUE.
                                   031A     ;
                                   031A     ; CALLING SEQUENCE:
                                   031A     ;
                                   031A     ;       JUMPED TO FROM AN FDT ROUTINE
                                   031A     ;
                                   031A     ; INPUT PARAMETERS:
                                   031A     ;
                                   031A     ;       R3      ADDRESS OF I/O PACKET
                                   031A     ;       R5      ADDRESS OF UCB
                                   031A     ;
                                   031A     ; OUTPUT PARAMETERS:
                                   031A     ;
                                   031A     ;       NONE
                                   031A     ;--
                                   031A
                                   031A     QUE_PKT:
                                   031A             DSBINT  UCB$B_FIPL(R5)          ; RAISE IPL TO FORK LEVEL
       08 64 A5   08       E2      0321             BBSS    #UCB$V_BSY,UCB$W_STS(R5),10$   ; SET BUSY AND SEE IF IT WAS SET
         00000000'GF       16      0326             JSB     G^IOC$INITIATE          ; NOT BUSY, INITIATE FUNCTION
                  08       11      032C             BRB     20$
                                   032E
        52   00AC C5       DE      032E     10$:    MOVAL   UCB$L_INQFL(R5),R2      ; GET ADDRESS OF I/O QUEUE LISTHEAD
         00000000'GF       16      0333             JSB     G^EXE$INSERTIRP         ; INSERT IN QUEUE BY PRIORITY
                                   0339
                                   0339     20$:    ENBINT                          ; LOWER IPL
         00000000'GF       17      033C             JMP     G^EXE$QIORETURN         ; RETURN FROM QIO
```

```
                         0342          .SBTTL  STARTIO - MAIN DRIVER ENTRY POINT
                         0342
                         0342  ;++
                         0342  ; FUNCTIONAL DESCRIPTION:
                         0342  ;
                         0342  ;       THIS ROUTINE IS THE MAIN DRIVER ENTRY POINT.  IT STARTS THE I/O,
                         0342  ;       WAITS FOR AN INTERRUPT, COMPLETES THE I/O, AND STARTS THE NEXT ONE.
                         0342  ;
                         0342  ; CALLING SEQUENCE:
                         0342  ;
                         0342  ;       CALLED THROUGH THE DRIVER DISPATCH TABLE
                         0342  ;
                         0342  ; INPUT PARAMETERS:
                         0342  ;
                         0342  ;       R3      ADDRESS OF I/O PACKET
                         0342  ;       R5      ADDRESS OF UCB
                         0342  ;
                         0342  ; OUTPUT PARAMETERS:
                         0342  ;
                         0342  ;       NONE
                         0342  ;--
                         0342
                         0342          .ENABL  LSB
                         0342  STARTIO:
                         0342
                         0342          ASSUME  IRP$S_FCODE EQ 6
52   20 A3   C0 8F   8B  0342          BICB3   #^XC0,IRP$W_FUNC(R3),R2 ; GET FUNCTION CODE
                         0348
                         0348          ; DISPATCH TO APPROPRIATE ROUTINE
FD42 CF   07   52    3A  0348          LOCC    R2,#IOFCTBLN,IOFCTBL    ; LOCATE FUNCTION CODE IN TABLE
      51   24 A5     D0  034E          MOVL    UCB$L_CRB(R5),R1        ; GET POINTER TO CRB IN R1
                         0352          CASE    TYPE=B,SRC=R0,DISPLIST=<-
                         0352                  STRT_NXT_REQ,-          ; INVALID FUNCTION
                         0352                  STOP,-                  ; STOP
                         0352                  START_DATA,-            ; START DATA (PHYSICAL)
                         0352                  START_DATA,-            ; START DATA
                         0352                  SET_CLOCK,-             ; SET CLOCK (PHYSICAL)
                         0352                  SET_CLOCK,-             ; SET CLOCK
                         0352                  INITIALIZE,-            ; INITIALIZE
                         0352                  >
                         0364
                         0364          ; FALL THROUGH TO ...
                         0364
                         0364  ;
                         0364  ;       S T A R T    M I C R O P R O C E S S O R
                         0364  ;
                         0364          ; NOTE:  THIS QIO COMES HERE DIRECTLY FROM THE FDT ROUTINE.
                         0364          ; THEREFORE R4 POINTS TO LPA-11 CSR.
                         0364          ; CHECK FOR VALID MICROCODE BEFORE STARTING MICROPROCESSOR
                         0364          ASSUME  LA$M_MCVALID EQ 1
                         0364          DSBINT  #31                     ; DON'T ALLOW INTERRUPTS (LIKE PWRFAIL)
     03 44 A5        E8  036A          BLBS    UCB$L_DEVDEPEND(R5),10$ ; BRANCH IF MICROCODE IS VALID
         0085        31  036E          BRW     MCNVALID                ; BRANCH IF MICROCODE IS NOT VALID
                         0371  10$:
                         0371
                         0371          ; ACTUALLY START MICROPROCESSOR
      8800 8F        B0  0371          MOVW    #LA_CISR_M_RUN!LA_CISR_M_ENA,-  ; SET RUN AND ENABLE
```

```
                64        0375   916                    LA_CISR(R4)                   ; ARBITRATION BITS
                          0376   917            ENBINT                                ; ALLOW INTERRUPTS
                          0379   918
                          0379   919            ; WAIT FOR AT LEAST 1 MICROSECOND BEFORE ENABLING INTERRUPTS
                          0379   920            TIMEDWAIT TIME=#1                     ; 1 10MS WAIT LOOP
                          0397   921
                          0397   922            DSBINT  #31                           ; CHECK FOR VALID MICROCODE AGAIN
          55 44 A5   E9   039D   923            BLBC    UCB$L_DEVDEPEND(R5),MCNVALID  ; BRANCH IF MICROCODE NOT VALID
     64    0040 8F   A8   03A1   924            BISW    #LA_CISR_M_IE,LA_CISR(R4)   ; ENABLE READY IN INTERRUPTS
  02 A4    0040 8F   A8   03A6   925            BISW    #LA_COSR_M_IE,LA_COSR(R4)   ; ENABLE READY OUT INTERRUPTS
                59   11   03AC   926            BRB     WAIT                          ; WAIT FOR INTERRUPT
                          03AE   927
                          03AE   928    ;
                          03AE   929    ;       S E T   C L O C K
                          03AE   930    ;
                          03AE   931    SET_CLOCK:
0164 C5      38 A3   7D   03AE   932            MOVQ    IRP$L_MEDIA(R3),UCB$W_RDA(R5)   ; BUILD RDA IN UCB
                0E   11   03B4   933            BRB     RDA_IN_UCB
                          03B6   934
                          03B6   935    ;
                          03B6   936    ;       S T A R T   D A T A
                          03B6   937    ;
                          03B6   938    START_DATA:
              00C1   30   03B6   939            BSBW    SDATA                         ; PREPARE FOR START DATA
             70 50   E9   03B9   940            BLBC    R0,DONE                       ; ERROR
                06   11   03BC   941            BRB     RDA_IN_UCB
                          03BE   942
                          03BE   943    ;
                          03BE   944    ;       S T O P
                          03BE   945    ;
                          03BE   946    STOP:
                          03BE   947            ; RDA IS IN SIP (FROM WHEN REQUEST WAS STARTED)
                          03BE   948            ASSUME  SIP$W_MODE EQ 0
0164 C5      48 B3   B0   03BE   949            MOVW    @IRP$L_SIP(R3),UCB$W_RDA(R5)  ; COPY RDA INTO UCB
                          03C4   950
                          03C4   951    RDA_IN_UCB:
                          03C4   952            ; SET CLOCK, START DATA, AND STOP COME HERE.  THE RDA IS IN UCB$W_RDA.
                          03C4   953            ; GET 18 BIT UNIBUS ADDRESS OF RDA
   52      00A0 C5   D0   03C4   954            MOVL    UCB$L_RDABA(R5),R2
                13   11   03C9   955            BRB     COMMON
                          03CB   956
                          03CB   957    ;
                          03CB   958    ;       I N I T I A L I Z E
                          03CB   959    ;
                          03CB   960    INITIALIZE:
                          03CB   961            ; INITIALIZE IS THE ONLY FUNCTION WHERE THE RDA IS IN THE PROCESS
                          03CB   962            ; ADDRESS SPACE.  MOVE RDA DESCRIPTOR FROM IRP TO UCB.
  78 A5      2C A3   7D   03CB   963            MOVQ    IRP$L_SVAPTE(R3),UCB$L_SVAPTE(R5)
                          03D0   964
                          03D0   965            ; SET UP MAP REGISTERS
             37 A1   94   03D0   966            CLRB    CRB$L_INTD+VEC$B_DATAPATH(R1)  ; USE DIRECT DATAPATH
              01DC   30   03D3   967            BSBW    SETMAPREG                     ; REQUEST AND LOAD UBA MAP REGISTERS
             53 50   E9   03D6   968            BLBC    R0,DONE                       ; ALLOCATION FAILURE
             34 A1   D0   03D9   969            MOVL    CRB$L_INTD+VEC$W_MAPREG(R1),-  ; SAVE ALLOCATED MAP REGISTER
             40 A3        03DC   970                    IRP$L_RDAMAPREG(R3)      ; INFO. IN IRP.
                          03DE   971
                          03DE   972    COMMON: ; COMMON FUNCTION PROCESSING.  INITIALIZE, SET CLOCK, START
```

```
                           03DE              ; DATA, AND STOP ALL COME HERE.  R2 CONTAINS 18 BIT UNIBUS ADDRESS
                           03DE              ; OF RDA.
                           03DE
                           03DE              ; GET POINTER TO LPA-11 DEVICE REGISTERS
                           03DE              ASSUME  IDB$L_CSR EQ 0
          54   2C B1   D0  03DE              MOVL    @CRB$L_INTD+VEC$L_IDB(R1),R4    ; GET PTR TO 1ST DEVICE REGISTER
                           03E2
                           03E2              ; BUILD WORD TO LOAD INTO LA_CISR IN R1
     51   52   F2 8F   78  03E2              ASHL    #-14,R2,R1                      ; PUT HIGH TWO BITS INTO POSITION IN R1
            51    03   AA  03E7              BICW    #3,R1                           ; CLEAR LOW TWO BITS
                  51   B6  03EA              INCW    R1                              ; SET GO BIT
                           03EC
                           03EC              ; CHECK FOR VALID MICROCODE, LOAD LPA-11 REGISTERS, AND THEN WAIT
                           03EC              ; FOR INTERRUPT (THIS ALSO CHECKS FOR POWERFAIL)
                           03EC              DSBINT  #31                             ; DON'T ALLOW INTERRUPTS (LIKE PWRFAIL)
            0A 44 A5   E8  03F2              BLBS    UCB$L_DEVDEPEND(R5),LOAD  ; BRANCH IF MICROCODE IS VALID
                           03F6
                           03F6  MCNVALID:   ; MICROCODE IS NOT VALID - COMPLETE REQUEST WITH ERROR
                           03F6              ENBINT                                  ; ALLOW INTERRUPTS
       50    035C 8F   3C  03F9              MOVZWL  #SS$_MCNOTVALID,R0              ; ERROR CODE
                  2C   11  03FE              BRB     DONE                            ; COMPLETE REQUEST
                           0400
                           0400  LOAD:       ; LOAD LPA-11 REGISTERS
           04 A4  52   B0  0400              MOVW    R2,LA_RDA(R4)                   ; LOAD UNIBUS ADDRESS OF RDA
              64  51   A8  0404              BISW2   R1,LA_CISR(R4)                  ; GO!
                           0407
                           0407  WAIT:       ; WAIT FOR INTERRUPT
                           0407              WFIKPCH TIMEOUT,#2                      ; WAIT FOR READY IN INTERRUPT.
                           0411                                                      ; READY OUT INTERRUPTS DON'T COME HERE.
                           0411                                                      ; (GO TO 'TIMEOUT' ON TIMEOUT OR
                           0411                                                      ; POWERFAIL)
                           0411              IOFORK                                  ; FORK TO DRIVER LEVEL
          53    58 A5  D0  0417              MOVL    UCB$L_IRP(R5),R3                ; GET ADDRESS OF CURRENT I/O PACKET
                  14   13  041B              BEQL    STRT_NXT_REQ                    ; THERE IS NONE - ALREADY HANDLED
                58 A5  D4  041D              CLRL    UCB$L_IRP(R5)                   ; CLEAR CURRENT I/O PACKET
                  23   10  0420              BSBB    SETCHAR                         ; SET CHARACTERISTICS IF APPROPRIATE
                           0422
                           0422              ; COPY LPA REGISTERS FROM INTERRUPT SAVE AREA TO COMMON SAVE AREA
  00E4 C5    00F4 C5   7D  0422              MOVQ    UCB$W_RISAVE(R5),UCB$L_REGSAVE(R5)
                           0429
             50   01   3C  0429              MOVZWL  S^#SS$_NORMAL,R0                ; SUCCESS STATUS
                           042C
                           042C  DONE:       ; REQUESTS COME HERE WHEN DONE WITH STATUS IN R0
                  51   D4  042C              CLRL    R1
                00D9   30  042E              BSBW    REQ_COMPLETE
                           0431
                           0431  STRT_NXT_REQ:   ; START NEXT REQUEST
       53    00AC D5   0F  0431              REMQUE  @UCB$L_INQFL(R5),R3             ; GET NEXT I/O PACKET IN QUEUE
                  06   1D  0436              BVS     60$                             ; THERE ISN'T ONE
         00000000'GF   17  0438              JMP     G^IOC$INITIATE
    64 A5    0100 8F   AA  043E  60$:        BICW    #UCB$M_BSY,UCB$W_STS(R5)  ; CLEAR UNIT BUSY
                       05  0444              RSB
                           0445
                           0445              .DSABL  LSB
```

```
                              0445              .SBTTL  SETCHAR - SET CHARACTERISTICS
                              0445
                              0445      ;++
                              0445      ; FUNCTIONAL DESCRIPTION:
                              0445      ;
                              0445      ;       THIS ROUTINE SETS DEVICE DEPENDENT CHARACTERISTICS AFTER THE
                              0445      ;       SUCCESSFUL COMPLETION OF AN INITIALIZE OR SET CLOCK QIO.
                              0445      ;       FOR INITIALIZE, THE CONFIGURATION BITS ARE SET.  FOR SET CLOCK
                              0445      ;       THE CLOCK RATE AND PRESET ARE STORED IF CLOCK A WAS SET.
                              0445      ;
                              0445      ; CALLING SEQUENCE:
                              0445      ;
                              0445      ;       BSBW/B
                              0445      ;
                              0445      ; INPUT PARAMETERS:
                              0445      ;
                              0445      ;       R3      ADDRESS OF IRP
                              0445      ;       R5      ADDRESS OF UCB
                              0445      ;
                              0445      ; IMPLICIT INPUTS:
                              0445      ;
                              0445      ;       THE CHARACTERISTICS ARE IN OFFSETS IRP$L_MEDIA THROUGH
                              0445      ;       IRP$L_MEDIA+5 OF THE I/O PACKET
                              0445      ;
                              0445      ; OUTPUT PARAMETERS:
                              0445      ;
                              0445      ;       NONE
                              0445      ;
                              0445      ; SIDE EFFECTS:
                              0445      ;
                              0445      ;       RO,R2 ARE NOT PRESERVED
                              0445      ;--
                              0445
                              0445      SETCHAR:
                              0445              ASSUME  IRP$S_FCODE EQ 6
52   20 A3    C0 8F     8B    0445              BICB3   #^XCO,IRP$W_FUNC(R3),R2  ; GET I/O FUNCTION CODE
                              044B
                              044B              ; IS IT INITIALIZE?
         04     52      91    044B              CMPB    R2,#IO$_INITIALIZE
                09      12    044E              BNEQ    10$                             ; NO
     03    38 A3        F0    0450              INSV    IRP$L_MEDIA(R3),#LA$V_CONFIG,-   ; YES, STORE CONFIGURATION
        44 A5   0A            0454                      #LA$S_CONFIG,UCB$L_DEVDEPEND(R5) ; BITS
                20      11    0457              BRB     30$
                              0459
                              0459      10$:    ; IS IT A SET CLOCK (EITHER ONE)
         37     52      91    0459              CMPB    R2,#IO$_SETCLOCK
                05      13    045C              BEQL    20$                             ; YES
         05     52      91    045E              CMPB    R2,#IO$_SETCLOCKP
                16      12    0461              BNEQ    30$                             ; NO
                              0463
                              0463      20$:    ; IT'S A SET CLOCK.  ONLY SET CHARACTERISTICS IF CLOCK A WAS SET
     11 38 A3   04      E0    0463              BBS     #4,IRP$L_MEDIA(R3),30$  ; BRANCH IF CLOCK B IS BEING SET
50   3A A3    FF 8F     78    0468              ASHL    #-1,IRP$L_MEDIA+2(R3),RO  ; GET CLOCK A RATE IN LOW BITS OF RO
          0D    50      F0    046E              INSV    RO,#LA$V_RATE,-          ; STORE RATE IN CHARACTERISTICS
        44 A5   03            0471                      #LA$S_RATE,UCB$L_DEVDEPEND(R5)
                              0474
                              0474              ASSUME LA$V_PRESET EQ 16
```

```
                         3C A3   B0  0474   1085          MOVW    IRP$L_MEDIA+4(R3),-      ; STORE PRESET
                         46 A5       0477   1086                  UCB$L_DEVDEPEND+2(R5)
                                     0479   1087
                                 05  0479   1088 30$:     RSB
```

```
                             047A          .SBTTL  SDATA - START DATA PROCESSING
                             047A
                             047A  ;++
                             047A  ; FUNCTIONAL DESCRIPTION:
                             047A  ;
                             047A  ;       THIS ROUTINE PERFORMS THE PROCESSING NECESSARY FOR START DATA.
                             047A  ;       IT ALLOCATES A BUFFERED DATAPATH (IF THE REQUEST IS A DEDICATED
                             047A  ;       MODE REQUEST), ALLOCATES AND LOADS MAP REGISTERS FOR THE USW,
                             047A  ;       BUFFERS, AND RCL AND BUILDS THE RDA FROM INFORMATION IN THE SIP.
                             047A  ;
                             047A  ; CALLING SEQUENCE:
                             047A  ;
                             047A  ;       BSBW    SDATA
                             047A  ;
                             047A  ; INPUT PARAMETERS:
                             047A  ;
                             047A  ;       R1      ADDRESS OF CRB
                             047A  ;       R3      ADDRESS OF IRP
                             047A  ;       R5      ADDRESS OF UCB
                             047A  ;
                             047A  ; OUTPUT PARAMETERS:
                             047A  ;
                             047A  ;       R0      COMPLETION CODE
                             047A  ;
                             047A  ; COMPLETION CODES:
                             047A  ;
                             047A  ;       SS$_NORMAL      NORMAL
                             047A  ;       SS$_INSFMAPREG  INSUFFICIENT MAP REGISTERS
                             047A  ;       SS$_INSFBUFDP   NO DATAPATHS AVAILABLE
                             047A  ;
                             047A  ; SIDE EFFECTS:
                             047A  ;
                             047A  ;       R2,R4 ARE DESTROYED
                             047A  ;--
                             047A
    54    48 A3     D0       047A  SDATA:  MOVL    IRP$L_SIP(R3),R4        ; GET PTR TO SECONDARY I/O PACKET
                             047E
                             047E          ; IF A DEDICATED MODE TRANSFER, REQUEST A BUFFERED DATAPATH
  13 64      03     E0       047E          BBS     #3,SIP$W_MODE(R4),10$   ; BRANCH IF MULTI-REQUEST MODE
                             0482
 00000000'GF        16       0482          JSB     G^IOC$REQDATAPNW        ; DEDICATED MODE - GET A BDP
    51    24 A5     D0       0488          MOVL    UCB$L_CRB(R5),R1        ; RESTORE POINTER TO CRB
          75 50     E9       048C          BLBC    R0,60$                  ; ALLOCATION FAILURE
          37 A1     89       048F          BISB3   CRB$L_INTD+VEC$B_DATAPATH(R1),-  ; SAVE DATAPATH NUMBER AND
  33 A4      20              0492                  #VEC$M_LWAE,SIP$B_BFR_DATAP(R4)  ; SET LONGWORD ACCESS BIT
                             0495
                             0495  10$:    ; ALLOCATE AND LOAD MAP REGISTERS FOR BUFFERS, USW, AND RCL
                             0495          ASSUME  SIP$L_BFR_SVAPT EQ SIP$L_USW_SVAPT+12  ; USW MUST BE FIRST!
                             0495          ASSUME  SIP$L_RCL_SVAPT EQ SIP$L_BFR_SVAPT+12  ; RCL MUST BE LAST!
       54    1C     C0       0495          ADDL    #SIP$L_USW_SVAPT,R4     ; POINT TO FIRST SVAPTE
             03     DD       0498          PUSHL   #3
                             049A
  78 A5      84     7D       049A  15$:    MOVQ    (R4)+,UCB$L_SVAPTE(R5)  ; LOAD SVAPTE, BOFF, BCNT
             0F     13       049E          BEQL    20$                     ; (ONLY IN CASE OF NO RCL - THIS
                             04A0                                          ; WORKS ONLY IF RCL INFO. IS LAST)
37 A1     03 A4     90       04A0          MOVB    3(R4),CRB$L_INTD+VEC$B_DATAPATH(R1)  ; LOAD DATAPATH #
          010A      30       04A5          BSBW    SETMAPREG               ; ALLOCATE AND LOAD MAP REGISTERS
```

```
                 55 50  E9 04A8              BLBC    R0,50$                          ; ALLOCATION FAILURE
           84    34 A1  D0 04AB              MOVL    CRB$L_INTD+VEC$W_MAPREG(R1),(R4)+  ; SAVE MAPREG, NUMREG
                 E8 6E  F5 04AF        20$:  SOBGTR  (SP),15$
                           04B2
                           04B2              ; NOW BUILD THE RDA
           54    48 A3  D0 04B2              MOVL    IRP$L_SIP(R3),R4                ; RESTORE POINTER TO BEGINNING OF SIP
         50    0164 C5  3E 04B6              MOVAW   UCB$W_RDA(R5),R0                ; POINT TO RDA IN UCB
           80      64   7D 04BB              MOVQ    SIP$W_MODE(R4),(R0)+            ; STORE MODE, BYTE COUNT, AND VALID
                           04BE                                                      ; BUFFER MASK IN RDA
         FA A0     02   A6 04BE              DIVW2   #2,-6(R0)                       ; CONVERT BYTE TO WORD COUNT IN RDA
                           04C2
                           04C2              ; INSERT USW ADDRESS
       FC A0     20 A4  B0 04C2              MOVW    SIP$W_USW_BOFF(R4),-4(R0)   ; BYTE OFFSET
FC A0   09  09   24 A4  F0 04C7              INSV    SIP$W_USW_MAPRE(R4),#9,#9,-4(R0)  ; PAGE NUMBER
                           04CE
                           04CE              ; NOW INSERT BUFFER ADDRESSES
           6E      07   D0 04CE              MOVL    #7,(SP)
           52    02 A4  3C 04D1              MOVZWL  SIP$W_BCNT(R4),R2               ; BUFFER LENGTH
           60    2C A4  3C 04D5              MOVZWL  SIP$W_BFR_BOFF(R4),(R0) ; BYTE OFFSET
    60  09 09    30 A4  F0 04D9              INSV    SIP$W_BFR_MAPRE(R4),#9,#9,(R0)   ; FIRST BUFFER ADDRESS
           50      04   C0 04DF              ADDL    #4,R0                           ; POINT TO SECOND BUFFER
       80  FC A0   52   C1 04E2        40$:  ADDL3   R2,-4(R0),(R0)+                 ; DO REMAINING 7 BUFFERS (ALWAYS CALC.
                 F8 6E  F5 04E7              SOBGTR  (SP),40$                        ; ALL 8 BUFFERS EVEN IF THERE AREN'T
                           04EA                                                      ; THAT MANY).
                           04EA
                           04EA              ; NOW STORE RCL ADDRESS IF THERE IS ONE
           80    38 A4  3C 04EA              MOVZWL  SIP$W_RCL_BOFF(R4),(R0)+   ; IF THERE IS NO RCL,
FC A0   09  09   3C A4  F0 04EE              INSV    SIP$W_RCL_MAPRE(R4),#9,#9,-4(R0)   ; THIS STORES A ZERO
           80    0C A4  7D 04F5              MOVQ    SIP$L_SLVDATA(R4),(R0)+ ; COPY REST OF RDA
           80    14 A4  7D 04F9              MOVQ    SIP$L_SLVDATA+8(R4),(R0)+
           50      01   3C 04FD              MOVZWL  S^#SS$_NORMAL,R0
                           0500
           5E      04   C0 0500        50$:  ADDL    #4,SP
                        05 0503              RSB
                           0504
                           0504        60$:  ; NO DATAPATH
       50    033C 8F    3C 0504              MOVZWL  #SS$_INSFBUFDP,R0
                        05 0509              RSB
```

```
                          050A          .SBTTL  REQUEST COMPLETE PROCESSING
                          050A  
                          050A  ;++
                          050A  ; FUNCTIONAL DESCRIPTION:
                          050A  ;
                          050A  ;       THIS ROUTINE RELEASES VARIOUS RESOURCES (UNLOCKS PAGES, RELEASES
                          050A  ;       MAP REGISTERS AND DATAPATH, AND DEALLOCATES SIP) BEFORE SENDING
                          050A  ;       AN I/O PACKET TO I/O POST PROCESSING.
                          050A  ;       THIS ROUTINE ALSO DOES SOME STUFF FOR ERROR LOGGING AND DIAGNOSTICS
                          050A  ;
                          050A  ; CALLING SEQUENCE:
                          050A  ;
                          050A  ;       BSBW    REQ_COMPLETE
                          050A  ;       BRW     REQ_COMPLETE
                          050A  ;
                          050A  ; INPUT PARAMETERS:
                          050A  ;
                          050A  ;       R0      FIRST LONGWORD OF I/O STATUS BLOCK
                          050A  ;       R1      SECOND LONGWORD OF I/O STATUS BLOCK
                          050A  ;               NOTE:   IF QIO IS A STOP, THEN STATUS IS ALREADY IN I/O PACKET
                          050A  ;       R3      ADDRESS OF I/O PACKET
                          050A  ;       R5      ADDRESS OF UCB
                          050A  ;
                          050A  ; OUTPUT PARAMETERS:
                          050A  ;
                          050A  ;       NONE
                          050A  ;--
                          050A  
                          050A  
                          050A  REQ_COMPLETE:
                3F    BB  050A          PUSHR   #^M<R0,R1,R2,R3,R4,R5>
 54   20 A3    C0 8F    8B  050C          BICB3   #^XC0,IRP$W_FUNC(R3),R4 ; GET FUNCTION CODE
          00EC C5    7C  0512          CLRQ    UCB$L_REGSAVE+8(R5)     ; CLEAR DATAPATH # AND REGISTER IN
                          0516                                          ; REGISTER SAVE AREA
                          0516  
                          0516          ; IF THIS IS A STOP REQUEST, THEN DON'T LOAD I/O STATUS
          54    03    91  0516          CMPB    #IO$_STOP,R4            ; STOP REQUEST?
                04    13  0519          BEQL    5$                      ; YES, DON'T LOAD STATUS
       38 A3    50    7D  051B          MOVQ    R0,IRP$L_IOST1(R3)      ; NO, LOAD IOSB
                          051F  
                          051F  5$:     ; GET POINTER TO CRB
       51    24 A5    D0  051F          MOVL    UCB$L_CRB(R5),R1
                          0523  
                          0523          ; IF THIS IS AN INITIALIZE QIO, RELEASE MAP REGISTERS POINTING TO RDA
          54    04    91  0523          CMPB    #IO$_INITIALIZE,R4      ; INITIALIZE?
                08    12  0526          BNEQ    10$                     ; NO
             40 A3    D0  0528          MOVL    IRP$L_RDAMAPREG(R3),-   ; GET STARTING MAP # AND NUMBER OF
             34 A1        052B                  CRB$L_INTD+VEC$W_MAPREG(R1)  ; REGISTERS AND MOVE INTO CRB
              0135    30  052D          BSBW    REL_MRDP                ; RELEASE THEM
                          0530  
                          0530  10$:    ; IF THIS WAS A START DATA OR STOP, GET POINTER TO SEC. I/O PACKET (SIP)
          54    38    91  0530          CMPB    #IO$_STARTDATA,R4       ; START DATA?
                0A    13  0533          BEQL    15$                     ; YES
          54    06    91  0535          CMPB    #IO$_STARTDATAP,R4      ; START DATA PHYSICAL?
                05    13  0538          BEQL    15$                     ; YES
          54    03    91  053A          CMPB    #IO$_STOP,R4            ; STOP?
                27    12  053D          BNEQ    30$                     ; NO
```

```
        54   48 A3  D0 053F  15$:    MOVL    IRP$L_SIP(R3),R4          ; GET POINTER TO SIP
                       0543
                       0543          ; RELEASE MAP REGISTERS FOR USW, DATA BUFFERS, AND RCL.
             24 A4  D0 0543          MOVL    SIP$W_USW_MAPRE(R4),-     ; STARTING MAP REGISTER # AND NUMBER
             34 A1     0546                  CRB$L_INTD+VEC$W_MAPREG(R1)   ; OF REGISTERS FOR USW.
                03  13 0548          BEQL    16$                       ; NONE
              0118  30 054A          BSBW    REL_MRDP                  ; RELEASE USW MAP REGISTERS
             30 A4  D0 054D  16$:    MOVL    SIP$W_BFR_MAPRE(R4),-     ; SAME FOR DATA BUFFERS, BUT ALSO
             34 A1     0550                  CRB$L_INTD+VEC$W_MAPREG(R1)   ; INCLUDE BUFFERED DP #, IF ANY
                03  13 0552          BEQL    18$                       ; NONE
              010E  30 0554          BSBW    REL_MRDP                  ; RELEASE MAP REGISTERS AND DATAPATH
             3C A4  D0 0557  18$:    MOVL    SIP$W_RCL_MAPRE(R4),-     ; SAME FOR RCL, IF THERE IS ONE
             34 A1     055A                  CRB$L_INTD+VEC$W_MAPREG(R1)
                03  13 055C          BEQL    20$                       ; NONE
              0104  30 055E          BSBW    REL_MRDP                  ; RELEASE RCL MAP REGISTERS
                       0561
                       0561  20$:    ; NOW UNLOCK PAGES FOR USW, DATA BUFFERS, AND RCL AND DEALLOCATE SIP.
           55   54  D0 0561          MOVL    R4,R5
                0C  10 0564          BSBB    UNLOCKF
                       0566
                       0566  30$:    ; DO ERROR LOGGING AND DIAGNOSTIC STUFF
                3F  BA 0566          POPR    #^M<R0,R1,R2,R3,R4,R5>
              037C  30 0568          BSBW    DODIAGERL
                       056B
                       056B          ; NOW QUEUE I/O PACKET FOR I/O POST PROCESSING
       00000000'GF  16 056B          JSB     G^COM$POST
                    05 0571          RSB
```

```
                      0572          .SBTTL  UNLOCK - UNLOCK PAGES AND DEALLOCATE SIP
                      0572
                      0572  ;++
                      0572  ; FUNCTIONAL DESCRIPTION:
                      0572  ;
                      0572  ;       THIS ROUTINE UNLOCKS PAGES WHICH WERE LOCKED FOR A DATA TRANSFER
                      0572  ;       AND DEALLOCATES THE SIP.  IT HAS TWO ENTRY POINTS: ONE SIMPLY
                      0572  ;       UNLOCKS THE PAGES; THE OTHER FORKS (USING THE SIP AS A FORK BLOCK)
                      0572  ;       BEFORE UNLOCKING THE PAGES.  PAGES ARE UNLOCKED FOR THE USW, THE
                      0572  ;       DATA BUFFERS, AND THE RCL.
                      0572  ;
                      0572  ; CALLING SEQUENCE:
                      0572  ;
                      0572  ;       BSBW    UNLOCK          (DOESN'T FORK)
                      0572  ;       BSBW    UNLOCKF         (FORKS)
                      0572  ;
                      0572  ; INPUT PARAMETERS:
                      0572  ;
                      0572  ;       R5      ADDRESS OF SIP
                      0572  ;
                      0572  ; OUTPUT PARAMEMTERS:
                      0572  ;
                      0572  ;       NONE
                      0572  ;
                      0572  ; SIDE EFFECTS:
                      0572  ;
                      0572  ;       R0 - R5 ARE NOT PRESERVED
                      0572  ;--
                      0572
                      0572  UNLOCKF:  ; FORK ENTRY POINT
      0B A5   06   90 0572          MOVB    #IPL$_QUEUEAST,FKB$B_FIPL(R5)  ; LOAD FORK IPL
                      0576          FORK
                      057C
                      057C  UNLOCK:  ; NO FORK ENTRY POINT
                      057C
                      057C          ; UNLOCK PAGES
              55   DD 057C          PUSHL   R5                      ; SAVE POINTER TO BEGINNING OF SIP
         55   1C   C0 057E          ADDL    #SIP$L_USW_SVAPT,R5     ; POINT TO FIRST SVAPTE
         54   03   D0 0581          MOVL    #3,R4                   ; LOOP 3 TIMES (USW, DATA BUFFERS, RCL)
                      0584
                      0584  10$:    ; UNLOCK NEXT AREA
         53   65   D0 0584          MOVL    (R5),R3                 ; GET SVAPTE
              19   13 0587          BEQL    20$                     ; NOTHING THERE
      51   04 A5   3C 0589          MOVZWL  4(R5),R1                ; GET BOFF
      52   06 A5   3C 058D          MOVZWL  6(R5),R2                ; GET BCNT
   51   01FF C142  9E 0591          MOVAB   511(R1)[R2],R1          ; COMBINE OFFSET AND COUNT AND ROUND
 51   51   F7 8F   78 0597          ASHL    #-VA$S_BYTE,R1,R1       ; CONVERT TO # OF PAGES (TO UNLOCK)
    00000000'GF    16 059C          JSB     G^MMG$UNLOCK            ; UNLOCK THEM
         55   0C   C0 05A2  20$:    ADDL    #12,R5                  ; POINT TO NEXT SET OF INFO.
           DC 54   F5 05A5          SOBGTR  R4,10$
                      05A8
                      05A8          ; NOW DEALLOCATE SIP
         50   8E   D0 05A8          MOVL    (SP)+,R0                ; GET POINTER TO BEGINNING OF SIP
    00000000'GF    16 05AB          JSB     G^EXE$DEANONPAGED
                   05 05B1          RSB
```

```
                  05B2          .SBTTL  SETMAPREG - ALLOCATE AND LOAD UBA MAP REGISTERS
                  05B2
                  05B2  ;++
                  05B2  ; FUNCTIONAL DESCRIPTION:
                  05B2  ;
                  05B2  ;       THIS ROUTINE ALLOCATES AND LOADS UBA MAPPING REGISTERS.
                  05B2  ;       IF MAPPING REGISTERS WERE PREALLOCATED THEN THE ALLOCATION IS FROM
                  05B2  ;       THE BITMAP IN THE UCB.  OTHERWISE THE ALLOCATION IS FROM THE BITMAP
                  05B2  ;       IN THE ADP.
                  05B2  ;
                  05B2  ; CALLING SEQUENCE:
                  05B2  ;
                  05B2  ;       BSBW    SETMAPREG
                  05B2  ;
                  05B2  ; INPUT PARAMETERS:
                  05B2  ;
                  05B2  ;       R1      POINTS TO CRB
                  05B2  ;       R5      POINTS TO UCB
                  05B2  ;
                  05B2  ; IMPLICIT INPUTS:
                  05B2  ;
                  05B2  ;       UCB$L_SVAPTE, UCB$W_BCNT, UCB$W_BOFF DESCRIBE THE AREA TO BE MAPPED
                  05B2  ;       UCB$L_PREALLOC IS NON-ZERO IF MAP REGISTERS WERE PREALLOCATED
                  05B2  ;       CRB$L_INTD+VEC$B_DATAPATH CONTAINS THE DATAPATH NUMBER TO USE
                  05B2  ;
                  05B2  ; OUTPUT PARAMETERS:
                  05B2  ;
                  05B2  ;       R0      CONTAINS A COMPLETION CODE (SEE BELOW)
                  05B2  ;       R2      CONTAINS 18 BIT STARTING UNIBUS ADDRESS OF AREA MAPPED
                  05B2  ;
                  05B2  ; IMPLICIT OUTPUTS:
                  05B2  ;
                  05B2  ;       CRB$L_INTD+VEC$W_MAPREG CONTAINS STARTING MAP REGISTER NUMBER
                  05B2  ;       CRB$L_INTD+VEC$B_NUMREG CONTAINS NUMBER OF MAPPING REGISTERS ALLOCATED
                  05B2  ;
                  05B2  ; COMPLETION CODES:
                  05B2  ;
                  05B2  ;       SS$_NORMAL      ALLOCATION WAS SUCCESSFUL
                  05B2  ;       SS$_INSFMAPREG  ALLOCATION FAILED (INSUFFICIENT MAP REGISTERS)
                  05B2  ;
                  05B2  ; SIDE EFFECTS:
                  05B2  ;
                  05B2  ;       NONE
                  05B2  ;
                  05B2  ;--
                  05B2  SETMAPREG:
                  05B2
                  05B2  ; If map registers were preallocated, then we call local subroutine
                  05B2  ;       ALLOC_LOCALMR to use some of preallocated registers.  Else we
                  05B2  ;       use normal system subroutine to allocate from central pool.
                  05B2
   00A8 C5   D5   05B2          TSTL    UCB$L_PREALLOC(R5)      ; ANY REGISTERS PREALLOCATED?
        04   13   05B6          BEQL    10$                     ; NO, PROCEED NORMALLY
        2D   10   05B8          BSBB    ALLOC_LOCALMR           ; Allocate from local pool.
        0A   11   05BA          BRB     20$                     ;  and branch around normal path.
                  05BC
                  05BC  10$:    ; ALLOCATE MAPPING REGISTERS
```

```
           00000000'GF   16  05BC  1385          JSB     G^IOC$ALOUBAMAP
             51   24 A5  D0  05C2  1386          MOVL    UCB$L_CRB(R5),R1          ; REFRESH R1 => CRB.
                             05C6  1387 20$:
                  18 50  E9  05C6  1388          BLBC    R0,50$                    ; ALLOCATION FAILURE
                             05C9  1389
                             05C9  1390          ; LOAD UNIBUS MAPPING REGISTERS
                     12  BB  05C9  1391          PUSHR   #^M<R1,R4>
           00000000'GF   16  05CB  1392          JSB     G^IOC$LOADUBAMAP
                     12  BA  05D1  1393          POPR    #^M<R1,R4>
                             05D3  1394
                             05D3  1395          ; SET UP STARTING UNIBUS ADDRESS OF AREA MAPPED
             52   7C A5  3C  05D3  1396          MOVZWL  UCB$W_BOFF(R5),R2         ; BYTE OFFSET IN PAGE (LOW 9 BITS)
 52    09    09   34 A1  F0  05D7  1397          INSV    CRB$L_INTD+VEC$W_MAPREG(R1),#9,#9,R2  ; HIGH 9 BITS
                             05DD  1398
                50    01  3C  05DD  1399          MOVZWL  S^#SS$_NORMAL,R0          ; SUCCESSFUL ALLOCATION
                          05  05E0  1400          RSB
                             05E1  1401
                             05E1  1402
                             05E1  1403 50$:     ; ALLOCATION FAILED
          50    0344 8F  3C  05E1  1404          MOVZWL  #SS$_INSFMAPREG,R0        ; INSUFFICIENT MAP REGISTERS
                         05  05E6  1405          RSB
```

```
                                      05E7  1407          .SBTTL  ALLOCATE UBA MAP REGISTERS FROM LOCAL POOL
                                      05E7  1408 ;+
                                      05E7  1409 ; ALLOC_LOCALMR
                                      05E7  1410 ;
                                      05E7  1411 ; THIS ROUTINE IS CALLED TO ALLOCATE UBA MAP REGISTERS AND TO MARK THE ALLOCATION
                                      05E7  1412 ; IN THE UBA MAP REGISTER ALLOCATION BITMAP MAINTAINED LOCALLY.
                                      05E7  1413 ;
                                      05E7  1414 ; INPUTS:
                                      05E7  1415 ;
                                      05E7  1416 ;        R5 = DEVICE UNIT UCB ADDRESS.
                                      05E7  1417 ;
                                      05E7  1418 ; OUTPUTS:
                                      05E7  1419 ;
                                      05E7  1420 ;        R0 = SUCCESS INDICATION.
                                      05E7  1421 ;-
                                      05E7  1422
                                      05E7  1423 ALLOC_LOCALMR:                                 ;ALLOCATE UBA MAP REGISTERS CRB SPECIFIED
                   7E     53     7D   05E7  1424         MOVQ    R3,-(SP)                      ; Save R3 and R4.
            53     7E A5         3C   05EA  1425         MOVZWL  UCB$W_BCNT(R5),R3             ;GET TRANSFER BYTE COUNT
            54     7C A5         3C   05EE  1426         MOVZWL  UCB$W_BOFF(R5),R4             ;GET BYTE OFFSET IN PAGE
      53    03FF C344            9E   05F2  1427         MOVAB   ^X3FF(R3)[R4],R3              ;CALCULATE HIGHEST RELATIVE BYTE AND ROUND
     53     53     F7 8F         78   05F8  1428         ASHL    #-9,R3,R3                     ;CALCULATE NUMBER OF MAP REGISTERS REQUIRED
                          50     D4   05FD  1429 5$:     CLRL    R0                            ;ASSUME ALLOCATION FAILURE
            51     24 A5         D0   05FF  1430         MOVL    UCB$L_CRB(R5),R1              ;GET ADDRESS OF CRB
            36 A1         53     90   0603  1431         MOVB    R3,CRB$L_INTD+VEC$B_NUMREG(R1) ;SET NUMBER OF MAP REGISTERS ALLOCATE
                          54     D4   0607  1432         CLRL    R4                            ;CLEAR STARTING BIT POSITION
            52     54     53     C1   0609  1433 10$:    ADDL3   R3,R4,R2                      ;CALCULATE HIGHEST BIT IN REQUIRED SCAN
            01F0 8F       52     B1   060D  1434         CMPW    R2,#496                       ;BEYOND END OF ALLOCATION BITMAP?
                          2D     14   0612  1435         BGTR    50$                           ;IF GTR YES
  54    0124 C5    20     54     EA   0614  1436         FFS     R4,#32,UCB$W_MRBITMAP(R5),R4  ;FIND A SET BIT
                          EC     13   061B  1437         BEQL    10$                           ;IF EQL BIT NOT FOUND
            52     54     53     C1   061D  1438         ADDL3   R3,R4,R2                      ;CALCULATE HIGH BIT FOR SUCCESSFUL ALLOCATIO
              34 A1       54     B0   0621  1439         MOVW    R4,CRB$L_INTD+VEC$W_MAPREG(R1) ;SAVE STARTING BIT NUMBER
  54    0124 C5    20     54     EB   0625  1440 20$:    FFC     R4,#32,UCB$W_MRBITMAP(R5),R4  ;FIND A CLEAR BIT
                   52     54     D1   062C  1441         CMPL    R4,R2                         ;ENOUGH SET BITS SCANNED OVER?
                          08     18   062F  1442         BGEQ    30$                           ;IF GEQ YES
        EE 0124 C5        54     E0   0631  1443         BBS     R4,UCB$W_MRBITMAP(R5),20$     ;IF SET, CONTINUE SCAN
                          D0     11   0637  1444         BRB     10$                           ;
            54     34 A1         3C   0639  1445 30$:    MOVZWL  CRB$L_INTD+VEC$W_MAPREG(R1),R4 ;RETRIEVE STARTING MAP REGISTER
                          06     10   063D  1446         BSBB    ALT_LOCALBITMAP               ;ALTER MAP REGISTER BITMAP
                          50     D6   063F  1447 40$:    INCL    R0                            ;SET SUCCESS INDICATOR
                   53     8E     7D   0641  1448 50$:    MOVQ    (SP)+,R3                      ;RESTORE REGISTERS
                                 05   0644  1449         RSB                                   ;
```

```
                            0645                .SBTTL  ALTER LOCAL UBA MAP REGISTER BITMAP
                            0645          ;+
                            0645          ; ALT_LOCALBITMAP
                            0645          ;
                            0645          ; THIS ROUTINE IS CALLED TO EITHER CLEAR OR SET A FIELD OF BITS IN THE UBA MAP
                            0645          ; REGISTER ALLOCATION BITMAP MAINTAINED LOCALLY IN THE UCB.
                            0645          ;
                            0645          ; INPUTS:
                            0645          ;
                            0645          ;       R0 = ALTERATION BIT MASK.
                            0645          ;       R1 = ADDRESS OF CRB.
                            0645          ;       R4 = STARTING MAP REGISTER NUMBER.
                            0645          ;       R5 => UCB
                            0645          ;
                            0645          ; OUTPUTS:
                            0645          ;
                            0645          ;       THE SPECIFIED BIT FIELD IN THE UBA MAP ALLOCATION BIT MAP IS EITHER SET
                            0645          ;       OR CLEARED DEPENDING ON THE STATE OF THE ALTERATION MASK.
                            0645          ;
                            0645          ;       R3 AND R4 ARE DESTROYED.
                            0645          ;-
                            0645
                            0645          ALT_LOCALBITMAP:
             53   36 A1   9A 0645                MOVZBL  CRB$L_INTD+VEC$B_NUMREG(R1),R3 ;GET NUMBER OF BITS TO ALTER
                  53  20  D1 0649          10$:   CMPL    #32,R3                         ;MORE THAN LONGWORD LEFT?
                      0F  18 064C                BGEQ    20$                            ;IF GEQ NO
 0124 C5   20     54  50  F0 064E                INSV    R0,R4,#32,UCB$W_MRBITMAP(R5) ;ALTER BITMAP WITH SUPPLIED PATTERN
                  54  20  C0 0655                ADDL    #32,R4                         ;UPDATE STARTING BIT POSITION
                  53  20  C2 0658                SUBL    #32,R3                         ;REDUCE NUMBER OF BITS TO ALTER
                      EC  11 065B                BRB     10$                            ;
 0124 C5   53     54  50  F0 065D          20$:   INSV    R0,R4,R3,UCB$W_MRBITMAP(R5) ;ALTER BITMAP WITH SUPPLIED PATTERN
                          05 0664                RSB                                    ;
```

```
                    0665       .SBTTL  REL_MRDP - RELEASE UBA MAP REGISTERS AND DATAPATH
                    0665
                    0665  ;++
                    0665  ; FUNCTIONAL DESCRIPTION:
                    0665  ;
                    0665  ;       THIS ROUTINE RELEASES UBA MAP REGISTERS AND A BUFFERED
                    0665  ;       DATAPATH IF ONE WAS ASSIGNED.  IF MAPPING REGISTERS
                    0665  ;       WERE PREALLOCATED, THEN THEY ARE RELEASED INTO THE BITMAP IN THE
                    0665  ;       UCB.  OTHERWISE, THEY ARE RELEASED INTO THE BITMAP IN THE ADP.
                    0665  ;       IN THE LATTER CASE AN ATTEMPT IS MADE TO CALL ANY DRIVERS WAITING
                    0665  ;       FOR MAP REGISTERS (ON THE ADP QUEUE).  BUFFERED DATAPATHS ARE
                    0665  ;       ALWAYS RELEASED INTO THE ADP BITMAP BECAUSE THEY ARE NOT PREALLOCATED.
                    0665  ;       ALSO, THE DATAPATH IS PURGED BEFURE IT IS RELEASED.
                    0665  ;       ALSO, THE DATAPATH NUMBER AND DATAPATH REGISTER ARE COPIED INTO
                    0665  ;       THE REGISTER SAVE AREA FOR DIAGNOSTICS AND ERROR LOGGING USE.
                    0665  ;
                    0665  ; CALLING SEQUENCE:
                    0665  ;
                    0665  ;       BSBW    REL_MRDP
                    0665  ;
                    0665  ; INPUT PARAMETERS:
                    0665  ;
                    0665  ;       R1      POINTS TO CRB
                    0665  ;       R3      POINTS TO IRP
                    0665  ;       R5      POINTS TO UCB
                    0665  ;
                    0665  ; IMPLICIT INPUTS:
                    0665  ;
                    0665  ;       UCB$L_PREALLOC IS NON-ZERO IF MAP REGISTERS WERE PREALLOCATED
                    0665  ;       CRB$L_INTD+VEC$W_MAPREG CONTAINS THE STARTING MAP REGISTER NUMBER
                    0665  ;       CRB$L_INTD+VEC$B_NUMREG CONTAINS NUMBER OF MAP REGISTERS TO RELEASE
                    0665  ;       CRB$L_INTD+VEC$B_DATAPATH CONTAINS THE DATAPATH NUMBER (ZERO MEANS
                    0665  ;               A BUFFERED DATAPATH WASN'T ALLOCATED).
                    0665  ;
                    0665  ; OUTPUT PARAMETERS:
                    0665  ;
                    0665  ;       NONE
                    0665  ;
                    0665  ; SIDE EFFECTS:
                    0665  ;
                    0665  ;       IF THERE IS A DATAPATH ERROR, THEN THE STATUS SS$_PARITY IS STORED
                    0665  ;       IN THE I/O PACKET.
                    0665  ;--
                    0665
                    0665  REL_MRDP:
          17    BB  0665           PUSHR   #^M<R0,R1,R2,R4>
          53    DD  0667           PUSHL   R3                      ; SAVE R3 SEPARATELY
     00A8 C5    D5  0669           TSTL    UCB$L_PREALLOC(R5)      ; REGISTERS PREALLOCATED?
          0B    13  066D           BEQL    10$                     ; NO
                    066F
                    066F           ; REGISTERS WERE PREALLOCATED SO SET UP TO ALTER BITMAP IN UCB.
 54    34 A1    3C  066F           MOVZWL  CRB$L_INTD+VEC$W_MAPREG(R1),R4  ; STARTING MAP REGISTER #
    50    00    D2  0673           MCOML   #0,R0                   ; ALTER PATTERN
          CD    10  0676           BSBB    ALT_LOCALBITMAP         ; Alter local bit map.
          0A    11  0678           BRB     20$
                    067A
                    067A  10$:     ; REGISTERS WERE NOT PREALLOCATED SO RETURN THEM TO ADP BITMAP
     00000000'GF    16  067A        JSB     G^IOC$RELMAPREG
      51   24 A5    D0  0680        MOVL    UCB$L_CRB(R5),R1        ; RESTORE POINTER TO CRB
                        0684
                        0684  20$:  ; RELEASE DATAPATH IF ONE WAS ALLOCATED
           53   8E  D0  0684        MOVL    (SP)+,R3                ; RESTORE R3 (POINTER TO IRP)
           05   00  EF  0687        EXTZV   #VEC$V_DATAPATH,#VEC$S_DATAPATH,- ; EXTRACT DATAPATH NUMBER
      52   37   A1      068A                CRB$L_INTD+VEC$B_DATAPATH(R1),R2  ; INTO R2
                23  13  068D        BEQL    30$                     ; NONE ALLOCATED
                        068F
                        068F        ; PURGE DATAPATH
                0C  BB  068F        PUSHR   #^M<R2,R3>              ; SAVE D.P. NUMBER AND IRP POINTER
     00000000'GF    16  0691        JSB     G^IOC$PURGDATAP         ; RETURNS STATUS IN R0, D.P. REG. IN R1
                0C  BA  0697        POPR    #^M<R2,R3>
             06 50  E8  0699        BLBS    R0,25$                  ; NO TRANSMISSION ERROR
 38 A3   01F4 8F    3C  069C        MOVZWL  #SS$_PARITY,IRP$L_IOST1(R3)     ; YES, RETURN ERROR STATUS
                        06A2
                        06A2  25$:  ; SAVE DATAPATH NUMBER AND CONTENTS OF DATAPATH REGISTER IN REGISTER
                        06A2        ; SAVE AREA
   00EC C5     52   D0  06A2        MOVL    R2,UCB$L_REGSAVE+8(R5)  ; SAVE DATAPATH NUMBER
   00F0 C5     51   D0  06A7        MOVL    R1,UCB$L_REGSAVE+12(R5) ; SAVE DATAPATH REGISTER
                        06AC
     00000000'GF    16  06AC        JSB     G^IOC$RELDATAP          ; RELEASE DATAPATH
                        06B2
                17  BA  06B2  30$:  POPR    #^M<R0,R1,R2,R4>
                    05  06B4        RSB
                    06B5          .SBTTL  READY IN INTERRUPT SERVICE
                    06B5
                    06B5  ;++
                    06B5  ; FUNCTIONAL DESCRIPTION:
                    06B5  ;
                    06B5  ;       THIS ROUTINE IS THE READY-IN INTERRUPT SERVICE ROUTINE.
                    06B5  ;       ASSUMING THE INTERRUPT WAS EXPECTED, IT CALLS THE DRIVER AT
                    06B5  ;       THE INTERRUPT WAIT ADDRESS AND THEN RETURNS.  UNEXPECTED
                    06B5  ;       INTERRUPTS ARE IGNORED BY RETURNING IMMEDIATELY.
                    06B5  ;
                    06B5  ; CALLING SEQUENCE:
                    06B5  ;
                    06B5  ;       JSB FROM INTERRUPT VECTOR IN CRB
                    06B5  ;
                    06B5  ; INPUT PARAMETERS:
                    06B5  ;
                    06B5  ;       NONE
                    06B5  ;
                    06B5  ; IMPLICIT INPUTS:
                    06B5  ;
                    06B5  ;       THE STACK ON ENTRY IS AS FOLLOWS:
                    06B5  ;
                    06B5  ;               0(SP)           ADDRESS OF IDB ADDRESS
                    06B5  ;       4(SP) - 24(SP)          SAVED R0 - R5
                    06B5  ;               28(SP)          INTERRUPT PC
                    06B5  ;               32(SP)          INTERRUPT PSL
                    06B5  ;
                    06B5  ; OUTPUT PARAMETERS:
                    06B5  ;
                    06B5  ;       NONE
                    06B5  ;--
                    06B5
                    06B5
                    06B5  LA$RDYININTSV::
      53   9E   D0  06B5          MOVL    @(SP)+,R3               ; GET ADDRESS OF IDB
                    06B8          ASSUME  IDB$L_CSR+4 EQ IDB$L_OWNER
      54   63   7D  06B8          MOVQ    IDB$L_CSR(R3),R4        ; CSR -> R4;   UCB -> R5
                    06BB
1E 64 A5   01   E5  06BB          BBCC    #UCB$V_INT,UCB$W_STS(R5),INTEXIT  ; IF CLR, INT. NOT EXPECTED
                    06C0
                    06C0          ; COPY LPA-11 REGISTERS INTO READY-IN INTERRUPT SAVE AREA
  00F4 C5  64   B0  06C0          MOVW    LA_CISR(R4),UCB$W_RISAVE(R5)
00F6 C5  02 A4  B0  06C5          MOVW    LA_COSR(R4),UCB$W_RISAVE+2(R5)
00F8 C5  04 A4  B0  06CB          MOVW    LA_RDA(R4),UCB$W_RISAVE+4(R5)
00FA C5  06 A4  B0  06D1          MOVW    LA_MAINT(R4),UCB$W_RISAVE+6(R5)
                    06D7
      53 10 A5  7D  06D7          MOVQ    UCB$L_FR3(R5),R3        ; RESTORE DRIVER CONTEXT
         0C B5  16  06DB          JSB     @UCB$L_FPC(R5)          ; CALL DRIVER AT INTERRUPT WAIT ADDRESS
                    06DE
                    06DE  INTEXIT:
      50   8E   7D  06DE          MOVQ    (SP)+,R0                ; RESTORE REGISTERS
      52   8E   7D  06E1          MOVQ    (SP)+,R2
      54   8E   7D  06E4          MOVQ    (SP)+,R4
                02  06E7          REI
```

```
                            06E8          .SBTTL  READY OUT INTERRUPT SERVICE
                            06E8
                            06E8  ;++
                            06E8  ; FUNCTIONAL DESCRIPTION:
                            06E8  ;
                            06E8  ;       THIS ROUTINE IS THE READY-OUT INTERRUPT SERVICE ROUTINE.
                            06E8  ;       AFTER RECEIVING THE INTERRUPT, THIS ROUTINE FORKS, DETERMINES
                            06E8  ;       THE CAUSE OF THE INTERRUPT, AND DISPATCHES TO AN APPROPRIATE
                            06E8  ;       ROUTINE.  THERE ARE BASICALLY FOUR CASES:
                            06E8  ;               1)  NO ERROR
                            06E8  ;                       A)  START REQUEST PROCESSED
                            06E8  ;                       B)  BUFFER FULL OR EMPTY
                            06E8  ;                       C)  BUFFER OVER/UNDERRUN
                            06E8  ;               2)  COMMAND ERROR
                            06E8  ;               3)  USER REQUEST ERROR (DURING A DATA TRANSFER)
                            06E8  ;               4)  FATAL HARDWARE ERROR
                            06E8  ;
                            06E8  ; CALLING SEQUENCE:
                            06E8  ;
                            06E8  ;       JSB FROM INTERRUPT VECTOR IN CRB
                            06E8  ;
                            06E8  ; INPUT PARAMETERS:
                            06E8  ;
                            06E8  ;       NONE
                            06E8  ;
                            06E8  ; IMPLICIT INPUTS:
                            06E8  ;
                            06E8  ;       THE STACK ON ENTRY IS AS FOLLOWS:
                            06E8  ;
                            06E8  ;                 0(SP)           ADDRESS OF IDB ADDRESS
                            06E8  ;       4(SP) - 24(SP)            SAVED R0 - R5
                            06E8  ;                28(SP)           INTERRUPT PC
                            06E8  ;                32(SP)           INTERRUPT PSL
                            06E8  ;
                            06E8  ; OUTPUT PARAMETERS:
                            06E8  ;
                            06E8  ;       NONE
                            06E8  ;--
                            06E8
                            06E8  LA$RDYOUTINTSV::
          53     9E     D0  06E8          MOVL    @(SP)+,R3               ; GET ADDRESS OF IDB
                            06EB          ASSUME  IDB$L_CSR+4 EQ IDB$L_OWNER
          54     63     7D  06EB          MOVQ    IDB$L_CSR(R3),R4        ; CSR -> R4;   UCB -> R5
                            06EE
                            06EE          ; COPY LPA-11 REGISTERS INTO READY-OUT INTERRUPT SAVE AREA
     00FC C5     64     B0  06EE          MOVW    LA_CISR(R4),UCB$W_ROSAVE(R5)
   00FE C5    02 A4     B0  06F3          MOVW    LA_COSR(R4),UCB$W_ROSAVE+2(R5)
   0100 C5    04 A4     B0  06F9          MOVW    LA_RDA(R4),UCB$W_ROSAVE+4(R5)
   0102 C5    06 A4     B0  06FF          MOVW    LA_MAINT(R4),UCB$W_ROSAVE+6(R5)
                            0705
              D6 AF     9F  0705          PUSHAB  INTEXIT                 ; ADDRESS TO RETURN TO AFTER FORK
      55    0084 C5     DE  0708          MOVAL   UCB$L_FORK0(R5),R5      ; HAVE TO USE DIFFERENT FORK BLOCK THAN
                            070D          FORK                            ; READY IN INTERRUPTS USE.
                            0713
      55    FF4C C5     DE  0713          MOVAL   -UCB$L_FORK0(R5),R5     ; RESTORE POINTER TO UCB
                            0718
                            0718          ; COPY LPA-11 REGISTERS FROM INTERRUPT SAVE AREA TO COMMON SAVE AREA
      00E4 C5   00FC C5   7D  0718               MOVQ    UCB$W_ROSAVE(R5),UCB$L_REGSAVE(R5)
                              071F
                              071F               ; GET CONTENTS OF CONTROL OUT STATUS REGISTER, AND MAINTENANCE REGISTER
                              071F               ; AND THEN ACKNOWLEGE INTERRUPT (WHICH ALLOWS THE NEXT READY OUT
                              071F               ; INTERRUPT TO OCCUR)
             50   02 A4   3C  071F               MOVZWL  LA_COSR(R4),R0             ; CONTROL OUT STATUS
             51   06 A4   3C  0723               MOVZWL  LA_MAINT(R4),R1            ; MAINTENANCE REGISTER
       02 A4    0080 8F   AA  0727               BICW2   #LA_COSR_M_RDY,LA_COSR(R4)  ; ACKNOWLEGE INTERRUPT
                              072D
                              072D               ; PUT BOTH LPA-11 REGISTERS INTO R1 TO BE USED AS SECOND
                              072D               ; LONGWORD OF IOSB IN CASE OF ERROR.
          51    51     10   78  072D               ASHL    #16,R1,R1                  ; PUT MAINT. REGISTER IN HIGH WORD
                51     50   B0  0731               MOVW    R0,R1                      ; PUT CONTROL OUT STATUS IN LOW WORD
                              0734
                              0734               ; GET USER # IN R2 AND DETERMINE IF THIS IS AN ERROR
  52    50   FFFFFFF8 8F   CB  0734               BICL3   #^XFFFFFFF8,R0,R2          ; GET USER INDEX IN R2
        50    50     F8 8F   78  073C               ASHL    #-8,R0,R0                  ; PUT STATUS ON LOW BYTE
                        50   95  0741               TSTB    R0                         ; ERROR?
                        03   19  0743               BLSS    ERROR                      ; YES
                      0079   31  0745               BRW     NO_ERROR                   ; NO
                              0748
                              0748      ;
                              0748      ;        E R R O R
                              0748      ;
                              0748      ERROR:   ; SOME SORT OF ERROR -  DETERMINE WHAT TYPE AND DISPATCH TO
                              0748               ; APPROPRIATE ROUTINE.  ERROR TYPE IS SPECIFIED BY FIELD
                              0748               ; LA COSR V_ERRTP WHICH HAS BEEN SHIFTED 8 BITS TO THE RIGHT IN R0
    02    50    02    05   ED  0748               CMPZV   #[A_COSR_V_ERRTP-8,#LA_COSR_S_ERRTP,R0,#2
                        3F   19  074D               BLSS    REQERR                     ; USER REQUEST ERROR
                        60   13  074F               BEQL    CMDERR                     ; COMMAND ERROR
                              0751
                              0751               ; FALL THROUGH TO ...
                              0751
                              0751      ;
                              0751      ;        F A T A L   H A R D W A R E   E R R O R
                              0751      ;
             50   0054 8F   3C  0751               MOVZWL  #SS$_CTRLERR,R0            ; STATUS
                        15   11  0756               BRB     COMP[_ALL_REQS
                              0758
                              0758      ;
                              0758      ;        T I M E O U T   O R   P O W E R F A I L
                              0758      ;
                              0758      TIMEOUT:  ; DEVICE TIMEOUT AND POWERFAIL COME HERE (AT DEVICE IPL).
                              0758               SETIPL  UCB$B_FIPL(R5)             ; LOWER TO FORK IPL
             50   0364 8F   3C  075C               MOVZWL  #SS$_POWERFAIL,R0          ; ASSUME POWERFAIL
                        51   D4  0761               CLRL    R1                         ; CLEAR SECOND LONGWORD OF IOSB
                        05   E0  0763               BBS     #UCB$V_POWER,-             ; BRANCH IF POWERFAIL
                  05 64 A5       0765                       UCB$W_STS(R5),COMPL_ALL_REQS
             50   022C 8F   3C  0768               MOVZWL  #SS$_TIMEOUT,R0            ; MUST BE TIMEOUT
                              076D
                              076D      COMPL_ALL_REQS:    ; COMPLETE ALL OUTSTANDING I/O REQUESTS
             53     58 A5   D0  076D               MOVL    UCB$L_IRP(R5),R3           ; GET CURRENT I/O REQUEST PACKET
                        06   13  0771               BEQL    10$                        ; THERE ISN'T ONE
                     58 A5   D4  0773               CLRL    UCB$L_IRP(R5)              ; CLEAR CURRENT I/O PACKET
                      FD91   30  0776               BSBW    REQ_COMPLETE               ; SEND IT TO REQUEST COMPLETE
                              0779
                              0779      10$:     ; NOW COMPLETE ALL OUTSTANDING DATA TRANSFER REQUESTS
```

```
           0259   30  0779          BSBW    COMPLETE_ALL
                      077C
                      077C          ; DO A DEVICE RESET (MASTER CLEAR) TO STOP MICROPROCESSOR
                      077C          DSBINT  UCB$B_DIPL(R5)          ; RAISE IPL TO DEVICE LEVEL
      64    4000 8F   B0  0783          MOVW    #LA_CISR_M_RESET,LA_CISR(R4)    ; RESET
                      0788          ENBINT                          ; LOWER IPL
                      078B
                      078B          ; REQUESTS ON THE INPUT QUEUE ARE STARTED IN THE NORMAL FASHION.
                      078B          ; HOWEVER, THEY ARE EXPECTED TO TIMEOUT.
           FCA3   31  078B          BRW     STRT_NXT_REQ            ; START NEXT REQUEST.
                      078E
                      078E
                      078E  ;
                      078E  ;       U S E R   R E Q U E S T   E R R O R
                      078E  ;
                      078E  REQERR: ; USER REQUEST ERROR
   53   0104 C542   D0  078E          MOVL    UCB$L_RQLIST(R5)[R2],R3 ; GET POINTER TO I/O PACKET
             1A   13  0794          BEQL    30$                     ; CAN HAPPEN IF STOP HAS BEEN QUEUED
                      0796                                          ; FOR THIS REQUEST
        0104 C542   D4  0796          CLRL    UCB$L_RQLIST(R5)[R2]    ; CLEAR SLOT
      50    A8 8F   91  079B          CMPB    #^O250,R0               ; STOPPED BY USW REQUEST?
             07   13  079F          BEQL    10$                     ; YES
   50   0334 8F   3C  07A1          MOVZWL  #SS$_DEVREQERR,R0       ; NO - ERROR.  LOAD STATUS RETURN
             05   11  07A6          BRB     20$
                      07A8
                      07A8  10$:    ; STOPPED BY USW REQUEST
      50    01   3C  07A8          MOVZWL  S^#SS$_NORMAL,R0        ; RETURN NORMAL STATUS
             51   D4  07AB          CLRL    R1                      ; CLEAR SECOND LONGWORD OF IOSB
           FD5A   30  07AD  20$:    BSBW    REQ_COMPLETE
             05   05  07B0  30$:    RSB
                      07B1
                      07B1  ;
                      07B1  ;       C O M M A N D   E R R O R
                      07B1  ;
                      07B1  CMDERR: ; COMMAND ERROR
      53    58 A5   D0  07B1          MOVL    UCB$L_IRP(R5),R3        ; GET POINTER TO CURRENT PACKET
            58 A5   D4  07B5          CLRL    UCB$L_IRP(R5)           ; CLEAR CURRENT PACKET ENTRY
   50   032C 8F   3C  07B8          MOVZWL  #SS$_DEVCMDERR,R0       ; STATUS RETURN
           FD4A   30  07BD          BSBW    REQ_COMPLETE
             05   05  07C0          RSB
                      07C1
                      07C1
                      07C1  ;
                      07C1  ;       N O   E R R O R
                      07C1  ;
                      07C1  NO_ERROR:  ; COME HERE IF THE INTERRUPT WAS NOT DUE TO AN ERROR.
                      07C1          ; THERE ARE THREE CASES:
                      07C1          ;       R0 = 0          START REQUEST PROCESSED
                      07C1          ;       R0 = 1          NORMAL BUFFER FULL/EMPTY
                      07C1          ;       R0 = 2          BUFFER OVER/UNDERRUN
                      07C1          ; NOTE:  WHEN WE GET HERE R0 HAS JUST BEEN TESTED.
                      07C1
             32   12  07C1          BNEQ    BFRFULL                 ; BUFFER FULL OR OVER/UNDERRUN
                      07C3
                      07C3  ;
                      07C3  ;       S T A R T   R E Q U E S T   P R O C E S S E D
                      07C3  ;
```

```
                            07C3  STARTREQ:
                            07C3                ; START REQUEST PROCESSED
           53    58 A5  D0  07C3                MOVL    UCB$L_IRP(R5),R3         ; GET POINTER TO I/O PACKET
           54    48 A3  D0  07C7                MOVL    IRP$L_SIP(R3),R4         ; GET POINTER TO SIP
              64    03  90  07CB                MOVB    #STOP_MODE,SIP$W_MODE(R4)  ; BUILD STOP RDA IN SIP
           01 A4    52  90  07CE                MOVB    R2,SIP$W_MODE+1(R4)      ; USER #
      0104 C542     53  D0  07D2                MOVL    R3,UCB$L_RQLIST(R5)[R2]  ; STORE ENTRY IN REQUEST LIST
                 58 A5  D4  07D8                CLRL    UCB$L_IRP(R5)            ; NO LONGER CURRENT PACKET
6C A5   0000000A'8F     C0  07DB                ADDL    I^#10,UCB$L_DUETIM(R5)   ; ADD 10 SECONDS TO DUE TIME TO PREVENT
                            07E3                                                 ; TIMEOUTS IN DEDICATED MODE WITH
                            07E3                                                 ; SLOW TRANSFERS.
                            07E3
                            07E3                ; NOW CHECK TO SEE IF THIS REQUEST HAS BEEN CANCELED
              50    2C  3C  07E3                MOVZWL  #SS$_ABORT,R0            ; ASSUME IT HAS
           64 A5    03  E0  07E6                BBS     #UCB$V_CANCEL,ULB$W_STS(R5),-    ; BRANCH IF IT HAS BEEN CANCELED
                    25      07EA                        QUEUE_STOP_REQ
                            07EB
                            07EB  10$:          ; NOW SIGNAL THAT REQUEST WAS STARTED
           54    3C A3  D0  07EB                MOVL    IRP$L_BFR_AST(R3),R4     ; USE BUFFER FULL AST ADDRESS
                    4E  10  07EF                BSBB    SIGNAL_BFR_FULL
                 1C 50  E9  07F1                BLBC    R0,QUEUE_STOP_REQ        ; ERROR
                        05  07F4                RSB
                            07F5
                            07F5
                            07F5
                            07F5  ;
                            07F5  ;             B U F F E R   F U L L   O R   O V E R / U N D E R R U N
                            07F5  ;
                            07F5  BFRFULL:
                            07F5                ; BUFFER FULL OR EMPTY (AND POSSIBLY OVER/UNDERRUN)
      53   0104 C542    D0  07F5                MOVL    UCB$L_RQLIST(R5)[R2],R3  ; GET POINTER TO I/O PACKET
                    12  13  07FB                BEQL    30$                      ; CAN HAPPEN IF STOP HAS BEEN QUEUED
           54    3C A3  D0  07FD                MOVL    IRP$L_BFR_AST(R3),R4     ; GET BUFFER FULL AST ADDRESS
              01    50  91  0801                CMPB    R0,#1                    ; BUFFER OVER/UNDERRUN?
                    04  13  0804                BEQL    20$                      ; NO
           54    40 A3  D0  0806                MOVL    IRP$L_OVR_AST(R3),R4     ; YES, GET BFR OVER/UNDERRUN AST ADDRESS
                    33  10  080A  20$:          BSBB    SIGNAL_BFR_FULL
                 01 50  E9  080C                BLBC    R0,QUEUE_STOP_REQ        ; ERROR
                        05  080F  30$:          RSB
```

```
                           0810       .SBTTL  QUEUE_STOP_REQ - QUEUE A STOP REQUEST
                           0810
                           0810 ;++
                           0810 ; FUNCTIONAL DESCRIPTION:
                           0810 ;
                           0810 ;      THIS ROUTINE TAKES AN I/O PACKET, CHANGES THE FUNCTION CODE TO
                           0810 ;      STOP, AND QUEUES THE PACKET TO THE DRIVER (AT THE HEAD OF THE
                           0810 ;      QUEUE).  IF THE DRIVER IS NOT BUSY, IT IS CALLED IMMEDIATELY.
                           0810 ;      IT IS ASSUMED THAT THE STOP RDA HAS ALREADY BEEN BUILT IN THE PACKET.
                           0810 ;      NOTE:  THIS ROUTINE MUST BE CALLED AT DRIVER FORK LEVEL.
                           0810 ;
                           0810 ; CALLING SEQUENCE:
                           0810 ;
                           0810 ;      BSBW    QUEUE_STOP_REQ   OR
                           0810 ;      BRW     QUEUE_STOP_REQ
                           0810 ;
                           0810 ; INPUT PARAMETERS:
                           0810 ;
                           0810 ;      R0      FIRST LONGWORD OF I/O STATUS BLOCK
                           0810 ;      R2      USER INDEX
                           0810 ;      R5      POINTER TO UCB
                           0810 ;
                           0810 ; OUTPUT PARAMETERS:
                           0810 ;
                           0810 ;      NONE
                           0810 ;--
                           0810
                           0810 QUEUE_STOP_REQ:
              3F     BB    0810        PUSHR   #^M<R0,R1,R2,R3,R4,R5>
53    0104 C542      D0    0812        MOVL    UCB$L_RQLIST(R5)[R2],R3 ; GET POINTER TO I/O PACKET
              22     13    0818        BEQL    40$                     ; PACKET ALREADY WENT AWAY
      0104 C542      D4    081A        CLRL    UCB$L_RQLIST(R5)[R2]    ; CLEAR SLOT
     20 A3    03     90    081F        MOVB    #IO$_STOP,IRP$W_FUNC(R3)  ; STORE STOP FUNCTION CODE IN IRP
     38 A3    50     D0    0823        MOVL    R0,IRP$L_IOST1(R3)      ; STORE STATUS CODE IN IOSB
           3C A3     D4    0827        CLRL    IRP$L_IOST2(R3)         ; CLEAR SECOND LONGWORD
                           082A
                           082A        ; REQUEUE PACKET IN FRONT IF I/O QUEUE (OR IF NOT BUSY, HANDLE IT NOW)
  08 64 A5    08     E2    082A        BBSS    #UCB$V_BSY,UCB$W_STS(R5),30$  ; SET BUSY; WAS IT ALREADY SET?
     00000000'GF     16    082F        JSB     G^IOC$INITIATE          ; NO, START DRIVER GOING
              05     11    0835        BRB     40$
                           0837
                           0837 30$:   ; DRIVER IS BUSY.  QUEUE PACKET
   00AC C5    63     0E    0837        INSQUE  IRP$L_IOQFL(R3),UCB$L_INQFL(R5)
              3F     BA    083C 40$:   POPR    #^M<R0,R1,R2,R3,R4,R5>
              05           083E        RSB
```

```
                        083F          .SBTTL  SIGNAL_BFR_FULL - SIGNAL BUFFER FULL (OR EMPTY) TO USER
                        083F
                        083F    ;++
                        083F    ; FUNCTIONAL DESCRIPTION:
                        083F    ;
                        083F    ;       THIS ROUTINE SIGNALS A USER PROCESS THAT A BUFFER HAS BEEN FILLED
                        083F    ;       OR EMPTIED.  SIGNALING IS DONE BY SETTING AN EVENT FLAG OR
                        083F    ;       ISSUING AN AST OR BOTH.  NOTE THAT THE SIGNALING IS DONE
                        083F    ;       AFTER A FORK HAS BEEN PERFORMED.
                        083F    ;
                        083F    ; CALLING SEQUENCE:
                        083F    ;
                        083F    ;       BSBB    SIGNAL_BFR_FULL
                        083F    ;
                        083F    ; INPUT PARAMETERS:
                        083F    ;
                        083F    ;       R3      ADDRESS OF I/O PACKET
                        083F    ;       R4      (USER) AST ADDRESS OR ZERO WHICH MEANS DON'T GIVE AN AST
                        083F    ;       R5      ADDRESS OF UCB
                        083F    ;
                        083F    ; IMPLICIT INPUTS:
                        083F    ;
                        083F    ;       VARIOUS FIELDS IN THE I/O PACKET
                        083F    ;
                        083F    ; OUTPUT PARAMETERS:
                        083F    ;
                        083F    ;       R0      COMPLETION CODE
                        083F    ;
                        083F    ; COMPLETION CODES:
                        083F    ;
                        083F    ;       SS$_NORMAL      NORMAL SUCCESSFUL COMPLETION
                        083F    ;       SS$_INSFMEM     INSUFFICIENT DYNAMIC MEMORY
                        083F    ;       SS$_EXQUOTA     EXCEEDED AST QUOTA
                        083F    ;
                        083F    ; SIDE EFFECTS:
                        083F    ;
                        083F    ;       R1 IS NOT PRESERVED
                        083F    ;--
                        083F
                        083F    SIGNAL_BFR_FULL:
            24    BB    083F            PUSHR   #^M<R2,R5>              ; SAVE REGISTERS HERE SO R5 CAN BE
            03    10    0841            BSBB    5$                      ; RESTORED AFTER FORK
            24    BA    0843            POPR    #^M<R2,R5>
                  05    0845            RSB
                        0846
                        0846    5$:     ; MAKE SURE THERE IS ENOUGH AST QUOTA TO ALLOCATE A FORK/AST BLOCK
   55     0C A3   3C    0846            MOVZWL  IRP$L_PID(R3),R5        ; GET PROCESS INDEX
 00000000'GF      DD    084A            PUSHL   G^SCH$GL_PCBVEC         ; PUSH ADDRESS OF PCB TABLE
   55   9E45      D0    0850            MOVL    @(SP)+[R5],R5           ; GET PCB ADDRESS
     50   1C      3C    0854            MOVZWL  #SS$_EXQUOTA,R0         ; ASSUME ERROR
       38 A5      B5    0857            TSTW    PCB$W_ASTCNT(R5)        ; ENOUGH AST QUOTA LEFT?
          1E      15    085A            BLEQ    10$                     ; NO
       38 A5      B7    085C            DECW    PCB$W_ASTCNT(R5)        ; YES, TAKE ONE AWAY
                        085F
                        085F            ; ALLOCATE A PACKET TO BE USED AS A FORK BLOCK AND AST CONTROL BLOCK
51    00C4 8F     3C    085F            MOVZWL  #IRP$C_LENGTH,R1        ; LENGTH = AN I/O PACKET
          53      DD    0864            PUSHL   R3                      ; SAVE R3
```

```
LA
Sy

IR
IR
IR
IR
IR
IR
IR
IR
IR
IR
IR
IR
IR
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LA
LE
LO
LO
LO
MA
MA
MC
MM
MM
NO
P1
P2
P3
P4
PC
PC
PR
```

```
            00000000'GF   16  0866   1936          JSB     G^EXE$ALONONPAGED          ; RETURNS POINTER TO PACKET IN R2
                 53  8E   D0  086C   1937          MOVL    (SP)+,R3                   ; RESTORE R3
                 09  50   E8  086F   1938          BLBS    R0,20$                     ; SUCCESSFUL ALLOCATION
          50   0124  8F   3C  0872   1939          MOVZWL  #SS$_INSFMEM,R0            ; ERROR - INSUFFICIENT DYNAMIC MEMORY
                 38  A5   B6  0877   1940          INCW    PCB$W_ASTCNT(R5)           ; ADD 1 BACK TO AST QUOTA
                      05  087A   1941 10$:         RSB                                ; ERROR RETURN
                          087B   1942
                          087B   1943 20$:         ; PUT SIZE AND TYPE INTO PACKET
                          087B   1944              ASSUME  IRP$B_TYPE EQ IRP$W_SIZE+2
  08 A2    000200C4  8F   D0  087B   1945          MOVL    #<DYN$C_ACB@16>+IRP$C_LENGTH,IRP$W_SIZE(R2)
                          0883   1946
                          0883   1947              ; NOW FORK (AND RETURN STATUS TO CALLER)
                          0883   1948              ASSUME FKB$B_FIPL EQ IRP$B_RMOD
          08 A2    06     90  0883   1949          MOVB    #IPL$_QUEUEAST,FKB$B_FIPL(R2)  ; SET FORK IPL
             55    52     D0  0887   1950          MOVL    R2,R5
             50    01     3C  088A   1951          MOVZWL  S^#SS$_NORMAL,R0           ; RETURN NORMAL STATUS TO CALLER
                          088D   1952              FORK
                          0893   1953
                          0893   1954              ; SET VARIOUS VALUES IN AST CONTROL BLOCK IN PREPARATION FOR
                          0893   1955              ; QUEUING AST
          51    0C A3     D0  0893   1956          MOVL    IRP$L_PID(R3),R1           ; SAVE PID FOR CALL TO SCH$POSTEF
          0C A5    51     D0  0897   1957          MOVL    R1,ACB$L_PID(R5)           ; PID
          10 A5    54     D0  089B   1958          MOVL    R4,ACB$L_AST(R5)           ; AST ADDRESS
                   0F     13  089F   1959          BEQL    30$                        ; NO AST
       0B A5    0B A3     90  08A1   1960          MOVB    IRP$B_RMOD(R3),ACB$B_RMOD(R5)    ; ACCESS MODE
       0B A5    40 8F     88  08A6   1961          BISB    #ACB$M_QUOTA,ACB$B_RMOD(R5)   ; SET AST QUOTA ACCOUNTING FLAG
       14 A5    14 A3     D0  08AB   1962          MOVL    IRP$L_ASTPRM(R3),ACB$L_ASTPRM(R5)  ; COPY AST PARAMETER
                          08B0   1963
                          08B0   1964 30$:         ; NOW POST EVENT FLAG IF SUBFUNCTION CODE SPECIFIES IT
             52    01     9A  08B0   1965          MOVZBL  #PRI$_IOCOM,R2             ; PRIORITY INCREMENT CLASS
       0A 20 A3    06     E1  08B3   1966          BBC     #IO$V_SETEVF,IRP$W_FUNC(R3),35$  ; BRANCH IF DON'T POST E.F.
          53    22 A3     9A  08B8   1967          MOVZBL  IRP$B_EFN(R3),R3           ; EVENT FLAG NUMBER
            00000000'GF   16  08BC   1968          JSB     G^SCH$POSTEF               ; POST EVENT FLAG
                          08C2   1969
                          08C2   1970 35$:         ; NOW EITHER GIVE AST OR DEALLOCATE BLOCK
                 10 A5    D5  08C2   1971          TSTL    ACB$L_AST(R5)              ; GIVE AST?
                    06    13  08C5   1972          BEQL    40$                        ; NO
            00000000'GF   17  08C7   1973          JMP     G^SCH$QAST                 ; YES
                          08CD   1974
                          08CD   1975 40$:         ; DON'T GIVE AST SO DEALLOCATE PACKET
          52    0C A5     3C  08CD   1976          MOVZWL  ACB$L_PID(R5),R2           ; BUT FIRST INCR. AST QUOTA
            00000000'GF   DD  08D1   1977          PUSHL   G^SCH$GL_PCBVEC            ; PUSH ADDRESS OF PCB TABLE
             52    9E42   D0  08D7   1978          MOVL    @(SP)+[R2],R2              ; GET PCB ADDRESS
                 38  A2   B6  08DB   1979          INCW    PCB$W_ASTCNT(R2)           ; INCR. QUOTA
             50    55     D0  08DE   1980          MOVL    R5,R0
            00000000'GF   17  08E1   1981          JMP     G^EXE$DEANONPAGED
```

```
                           08E7  1983          .SBTTL  DODIAGERL - DO DIAG. AND ERROR LOGGING STUFF
                           08E7  1984  ;++
                           08E7  1985  ; FUNCTIONAL DESCRIPTION:
                           08E7  1986  ;
                           08E7  1987  ;       THIS ROUTINE DOES THE FOLLOWING:
                           08E7  1988  ;               1)  CALLS THE DIAGNOSTIC BUFFER FILL ROUTINE WHICH COPIES
                           08E7  1989  ;                   THE REGISTER SAVE INFO. INTO A DIAGNOSTIC BUFFER IF ONE
                           08E7  1990  ;                   WAS SUPPLIED WITH THE REQUEST.
                           08E7  1991  ;               2)  IF THE I/O STATUS INDICATES A LOGGABLE ERROR, THEN
                           08E7  1992  ;                   THE ERROR IS LOGGED.  NOTE THAT THIS ROUTINE DOES THE
                           08E7  1993  ;                   PROCESSING NORMALLY DONE IN IOC$REQCOM SINCE THIS DRIVER
                           08E7  1994  ;                   DOESN'T CALL IOC$REQCOM.
                           08E7  1995  ;
                           08E7  1996  ; CALLING SEQUENCE:
                           08E7  1997  ;
                           08E7  1998  ;       BSBW    DODIAGERL
                           08E7  1999  ;
                           08E7  2000  ; INPUT PARAMETERS:
                           08E7  2001  ;
                           08E7  2002  ;       R0      FIRST LONGWORD OF I/O STATUS
                           08E7  2003  ;       R1      SECOND LONGWORD OF I/O STATUS
                           08E7  2004  ;       R3      ADDRESS OF IRP
                           08E7  2005  ;       R5      ADDRESS OF UCB
                           08E7  2006  ;
                           08E7  2007  ; IMPLICIT INPUTS:
                           08E7  2008  ;
                           08E7  2009  ;       VARIOUS FIELDS IN THE IRP AND UCB
                           08E7  2010  ;
                           08E7  2011  ; OUTPUT PARAMETERS:
                           08E7  2012  ;
                           08E7  2013  ;       NONE
                           08E7  2014  ;
                           08E7  2015  ; SIDE EFFECTS:
                           08E7  2016  ;
                           08E7  2017  ;       OFFSET UCB$W_FUNC IN THE UCB IS MODIFIED
                           08E7  2018  ;--
                           08E7  2019
                           08E7  2020  DODIAGERL:
                07    BB   08E7  2021          PUSHR   #^M<R0,R1,R2>
             58 A5    DD   08E9  2022          PUSHL   UCB$L_IRP(R5)           ; SAVE THIS 'CAUSE WE MODIFY IT
                           08EC  2023
   009A C5   20 A3    B0   08EC  2024          MOVW    IRP$W_FUNC(R3),UCB$W_FUNC(R5)   ; SAVE FUNCTION CODE
      58 A5     53    D0   08F2  2025          MOVL    R3,UCB$L_IRP(R5)        ; MAKE THIS IRP THE 'CURRENT' ONE
                           08F6  2026
                           08F6  2027          ; CALL DIAGNOSTIC BUFFER FILL ROUTINE
      00000000'GF     16   08F6  2028          JSB     G^IOC$DIAGBUFILL
                           08FC  2029
                           08FC  2030          ; CALL ERROR LOGGER IF WE HAVE A LOGGABLE ERROR
   022C 8F   38 A3    B1   08FC  2031          CMPW    IRP$L_IOST1(R3),#SS$_TIMEOUT    ; IS IT A TIMEOUT?
                08    12   0902  2032          BNEQ    10$                             ; NO
      00000000'GF     16   0904  2033          JSB     G^ERL$DEVICTMO                  ; YES, LOG TIMEOUT
                1E    11   090A  2034          BRB     40$
                           090C  2035
                           090C  2036  10$:    ; IS IT ANY OTHER LOGGABLE ERROR?
   0054 8F   38 A3    B1   090C  2037          CMPW    IRP$L_IOST1(R3),#SS$_CTRLERR    ; IS IT A FATAL HRDWRE ERROR?
                10    13   0912  2038          BEQL    30$                             ; YES
   0334 8F   38 A3    B1   0914  2039          CMPW    IRP$L_IOST1(R3),#SS$_DEVREQERR  ; IS IT A DEVICE REQUEST ERROR?
```

```
                08    13  091A  2040          BEQL    30$                   ; YES
       01F4 8F    38 A3    B1  091C  2041          CMPW    IRP$L_IOST1(R3),#SS$_PARITY     ; UBA PARITY ERROR?
                25    12  0922  2042          BNEQ    50$                             ; NO
                          0924  2043
        00000000'GF    16  0924  2044 30$:     JSB     G^ERL$DEVICERR                  ; LOG DEVICE ERROR
                          092A  2045
                          092A  2046 40$:     ; NOW FILL IN REST OF BUFFER LIKE IOC$REQCOM DOES
      1A 64 A5    02    E5  092A  2047          BBCC    #UCB$V_ERLOGIP,UCB$W_STS(R5),50$ ; CLEAR ERROR LOG IN PROGRESS
       52    0094 C5    D0  092F  2048          MOVL    UCB$L_EMB(R5),R2          ; GET ADDRESS OF ERROR MESSAGE BUFFER
      1A A2    64 A5    B0  0934  2049          MOVW    UCB$W_STS(R5),EMB$W_DV_STS(R2) ;INSERT FINAL DEVICE STATUS
     10 A2    0080 C5    B0  0939  2050          MOVW    UCB$B_ERTCNT(R5),EMB$B_DV_ERTCNT(R2)     ; INSERT ERROR COUNTERS
          12 A2    50    7D  093F  2051          MOVQ    R0,EMB$Q_DV_IOSB(R2)      ; INSERT I/O STATUS
                          0943  2052
        00000000'GF    16  0943  2053          JSB     G^ERL$RELEASEMB           ; RELEASE ERROR MESSAGE BUFFER
                          0949  2054
             58 A5 8ED0  0949  2055 50$:     POPL    UCB$L_IRP(R5)             ; RESTORE THIS LOCATION
                07    BA  094D  2056          POPR    #^M<R0,R1,R2>
                      05  094F  2057          RSB
```

```
                        0950  2059          .SBTTL  LA_REGDUMP - REGISTER DUMP ROUTINE
                        0950  2060  ;++
                        0950  2061  ; FUNCTIONAL DESCRIPTION
                        0950  2062  ;
                        0950  2063  ;       THIS ROUTINE WRITES THE SAVED REGISTERS INTO A BUFFER.  IT IS
                        0950  2064  ;       CALLED FROM THE ERROR LOGGING ROUTINE AND THE DIAGNOSTIC BUFFER
                        0950  2065  ;       FILL ROUTINE.
                        0950  2066  ;
                        0950  2067  ; CALLING SEQUENCE:
                        0950  2068  ;
                        0950  2069  ;       BSBW    LA_REGDUMP
                        0950  2070  ;
                        0950  2071  ; INPUT PARAMETERS:
                        0950  2072  ;
                        0950  2073  ;       R0      ADDRESS OF REGISTER SAVE BUFFER
                        0950  2074  ;       R5      ADDRESS OF UCB
                        0950  2075  ;
                        0950  2076  ; OUTPUT PARAMETERS:
                        0950  2077  ;
                        0950  2078  ;       NONE
                        0950  2079  ;
                        0950  2080  ; SIDE EFFECTS:
                        0950  2081  ;
                        0950  2082  ;       R1,R2 ARE NOT PRESERVED
                        0950  2083  ;--
                        0950  2084
                        0950  2085  LA_REGDUMP:
        80    06    D0  0950  2086          MOVL    #6,(R0)+                ; INSERT NUMBER OF REGISTERS INTO BFR
   51 00E4    C5    DE  0953  2087          MOVAL   UCB$L_REGSAVE(R5),R1    ; GET ADDRESS OF SAVE AREA
        52    04    D0  0958  2088          MOVL    #4,R2                   ; NUMBER OF LPA-11 REGISTERS
        80    81    3C  095B  2089  10$:    MOVZWL  (R1)+,(R0)+             ; COPY INTO BUFFER
           FA 52    F5  095E  2090          SOBGTR  R2,10$                  ; LOOP BACK
                        0961  2091
        60    61    7D  0961  2092          MOVQ    (R1),(R0)               ; COPY DATAPATH NUMBER AND REGISTER
                    05  0964  2093          RSB
```

```
                        0965          .SBTTL  CANCEL_IO - CANCEL I/O
                        0965
                        0965  ;++
                        0965  ; FUNCTIONAL DESCRIPTION:
                        0965  ;
                        0965  ;       THIS ROUTINE PERFORMS THE CANCEL I/O FUNCTION.  ONLY PACKETS
                        0965  ;       THAT HAVE A MATCHING CHANNEL INDEX AND PID ARE CANCELED.  FIRST, THE
                        0965  ;       CURRENT PACKET (IF THERE IS ONE) IS CANCELED BY SETTING THE CANCEL I/O
                        0965  ;       BIT IN THE UCB.  THEN ANY PACKETS ON THE INPUT QUEUE ARE CANCELED
                        0965  ;       BY SENDING THEM TO REQ_COMPLETE WITH A STATUS OF SS$_CANCEL.  THE
                        0965  ;       ONLY EXCEPTION IS THAT STOP QIO'S ARE NOT CANCELED.  FINALLY,
                        0965  ;       ONGOING DATA TRANSFERS ARE CANCELED BY SENDING THEM TO QUEUE_STOP_REQ
                        0965  ;       WITH A STATUS OF SS$_ABORT.
                        0965  ;
                        0965  ; CALLING SEQUENCE:
                        0965  ;
                        0965  ;       BSBW/B
                        0965  ;
                        0965  ; INPUT PARAMETERS:
                        0965  ;
                        0965  ;       R2      CHANNEL INDEX
                        0965  ;       R3      POINTER TO CURRENT I/O PACKET
                        0965  ;       R4      PCB ADDRESS
                        0965  ;       R5      POINTER TO UCB
                        0965  ;
                        0965  ; OUTPUT PARAMETERS:
                        0965  ;
                        0965  ;       NONE
                        0965  ;--
                        0965
                        0965  CANCEL_IO:
      00DC 8F     BB    0965          PUSHR   #^M<R2,R3,R4,R6,R7>
      57   52     D0    0969          MOVL    R2,R7                   ; CHANNEL INDEX
  54    60 A4     D0    096C          MOVL    PCB$L_PID(R4),R4        ; PUT PID IN R4
                        0970
                        0970          ; FIRST CANCEL CURRENT I/O PACKET IF THERE IS ONE
           53     D5    0970          TSTL    R3                      ; POINTER TO CURRENT PACKET
           08     13    0972          BEQL    10$                     ; NO CURRENT PACKET
           54     10    0974          BSBB    CANCELCK                ; CHECK CHANNEL AND PID
           04     12    0976          BNEQ    10$                     ; NOT A MATCH
     64 A5 08     A8    0978          BISW    #UCB$M_CANCEL,UCB$W_STS(R5)  ; SET CANCEL BIT
                        097C
                        097C  10$:    ; NOW CANCEL THE PACKETS ON THE INPUT QUEUE
  50  0830 8F     3C    097C          MOVZWL  #SS$_CANCEL,R0          ; STATUS
           51     D4    0981          CLRL    R1
  53  00AC C5     9E    0983          MOVAB   UCB$L_INQFL(R5),R3      ; GET POINTER TO QUEUE HEAD
      56   53     D0    0988          MOVL    R3,R6                   ; SAVE POINTER TO QUEUE HEAD
                        098B
                        098B  20$:    ; EXAMINE NEXT PACKET IN QUEUE
      53   63     D0    098B          MOVL    IRP$L_IOQFL(R3),R3      ; GET POINTER TO NEXT PACKET
      56   53     D1    098E          CMPL    R3,R6                   ; REACHED END OF QUEUE YET?
           1A     13    0991          BEQL    30$                     ; YES, DONE WITH THIS PHASE
           35     10    0993          BSBB    CANCELCK                ; CHECK CHANNEL AND PID
           F4     12    0995          BNEQ    20$                     ; NOT A MATCH, GET NEXT PACKET
     20 A3 03     91    0997          CMPB    #IO$_STOP,IRP$W_FUNC(R3)  ; DON'T CANCEL STOP REQUESTS
           EE     13    099B          BEQL    20$                     ; IT'S A STOP.  GET NEXT PACKET
  52    04 A3     D0    099D          MOVL    IRP$L_IOQBL(R3),R2      ; HAVE A PACKET TO REMOVE.  BACK UP
```

```
        53   00 B2   0F  09A1              REMQUE  @IRP$L_IOQFL(R2),R3      ; REMOVE PACKET FROM QUEUE
               FB62  30  09A5              BSBW    REQ_COMPLETE             ; SEND PACKET TO REQUEST COMPLETE
           53    52  D0  09A8              MOVL    R2,R3
                 DE  11  09AB              BRB     20$                      ; GET NEXT PACKET
                         09AD
                         09AD   30$:       ; NOW STOP ANY MATCHING DATA TRANSFER REQUESTS
           50    2C  3C  09AD              MOVZWL  #SS$_ABORT,R0            ; STATUS
                 52  D4  09B0              CLRL    R2
                         09B2
                         09B2   40$:       ; EXAMINE NEXT ENTRY IN REQUEST LIST
   53   0104 C542    D0  09B2              MOVL    UCB$L_RQLIST(R5)[R2],R3  ; GET POINTER TO PACKET
                 07  13  09B8              BEQL    50$                      ; EMPTY SLOT
                 0E  10  09BA              BSBB    CANCELCK                 ; CHECK CHANNEL AND PID
                 03  12  09BC              BNEQ    50$                      ; NOT A MATCH
               FE4F  30  09BE              BSBW    QUEUE_STOP_REQ           ; QUEUE A STOP REQUEST
        ED 52    08  F2  09C1   50$:       AOBLSS  #8,R2,40$                ; REPEAT FOR ALL 8 REQUESTS
                         09C5
           00DC 8F   BA  09C5              POPR    #^M<R2,R3,R4,R6,R7>
                     05  09C9              RSB
                         09CA
                         09CA
                         09CA
                         09CA
                         09CA   ; LOCAL SUBROUTINE TO CHECK FOR MATCHING CHANNEL AND PID
                         09CA   ;
                         09CA   ; INPUT:
                         09CA   ;          R3      POINTS TO I/O PACKET
                         09CA   ;          R4      CONTAINS PID
                         09CA   ;          R7      CONTAINS CHANNEL INDEX
                         09CA   ; OUTPUT:
                         09CA   ;          Z BIT   IS SET IF BOTH MATCH, CLEARED OTHERWISE
                         09CA
                         09CA   CANCELCK:
        54   0C A3   D1  09CA              CMPL    IRP$L_PID(R3),R4         ; CHECK PID
                 04  12  09CE              BNEQ    10$                      ; NO MATCH
        57   28 A3   B1  09D0              CMPW    IRP$W_CHAN(R3),R7        ; CHECK CHANNEL AND SET OR CLEAR Z BIT
                     05  09D4   10$:       RSB
```

```
                             09D5          .SBTTL  COMPLETE_ALL - COMPLETE ALL DATA TRANSFER REQUESTS
                             09D5
                             09D5   ;++
                             09D5   ; FUNCTIONAL DESCRIPTION:
                             09D5   ;
                             09D5   ;       THIS ROUTINE GOES THROUGH THE USER TABLE SENDING ALL CURRENT
                             09D5   ;       DATA TRANSFER REQUESTS TO REQ_COMPLETE.
                             09D5   ;
                             09D5   ; CALLING SEQUENCE:
                             09D5   ;
                             09D5   ;       BSBW    COMPLETE_ALL
                             09D5   ;
                             09D5   ; INPUT PARAMETERS:
                             09D5   ;
                             09D5   ;       R0      FIRST LONGWORD OF I/O STATUS BLOCK
                             09D5   ;       R1      SECOND LONGWORD OF I/O STATUS BLOCK
                             09D5   ;       R5      ADDRESS OF UCB
                             09D5   ;
                             09D5   ; OUTPUT PARAMETERS:
                             09D5   ;
                             09D5   ;       NONE
                             09D5   ;
                             09D5   ; SIDE EFFECTS:
                             09D5   ;
                             09D5   ;       R2,R3 ARE NOT SAVED
                             09D5   ;--
                             09D5
                             09D5   COMPLETE_ALL:
                             09D5
                  52    D4   09D5           CLRL    R2                      ; INITIALIZE INDEX INTO REQUEST LIST
                             09D7
                             09D7   20$:    ; DO NEXT ONE IN REQUEST LIST
     53   0104 C542    D0   09D7           MOVL    UCB$L_RQLIST(R5)[R2],R3 ; GET POINTER TO I/O PACKET
                  08    13   09DD           BEQL    30$                     ; NO REQUEST IN THIS SLOT
          0104 C542    D4   09DF           CLRL    UCB$L_RQLIST(R5)[R2]    ; CLEAR SLOT
               FB23    30   09E4           BSBW    REQ_COMPLETE            ; SEND IT TO REQUEST COMPLETE
          EC 52   08    F2   09E7   30$:    AOBLSS  #8,R2,20$               ; GO BACK FOR NEXT
                        05   09EB           RSB
```

```
                              09EC          .SBTTL  UNIT_INIT - LPA-11 UNIT INITIALIZATION
                              09EC  ;++
                              09EC  ; FUNCTIONAL DESCRIPTION:
                              09EC  ;
                              09EC  ;       THIS ROUTINE IS ENTERED WHEN THE DRIVER IS LOADED AND ON POWER
                              09EC  ;       RECOVERY.  ON DRIVER LOAD IT INITIALIZES THE UCB, OPTIONALLY
                              09EC  ;       PREALLOCATES MAP REGISTERS, AND ALLOCATES AND LOADS MAP REGISTERS
                              09EC  ;       TO PERMANENTLY MAP THE RDA IN THE UCB.  ON POWER RECOVERY, IT
                              09EC  ;       CLEARS THE MICROCODE VALID BIT, RELOADS THE MAP REGISTERS THAT
                              09EC  ;       MAP THE RDA IN THE UCB, AND THEN FORKS TO COMPLETE ALL ACTIVE
                              09EC  ;       REQUESTS WITH A STATUS OF SS$_POWERFAIL.
                              09EC  ;
                              09EC  ; CALLING SEQUENCE:
                              09EC  ;
                              09EC  ;       JSB     UNIT_INIT
                              09EC  ;
                              09EC  ; INPUT PARAMETERS:
                              09EC  ;
                              09EC  ;       R5      ADDRESS OF UCB
                              09EC  ;
                              09EC  ; OUTPUT PARAMETERS:
                              09EC  ;
                              09EC  ;       NONE
                              09EC  ;
                              09EC  ; SIDE EFFECTS:
                              09EC  ;
                              09EC  ;       R0 - R4 ARE NOT PRESERVED
                              09EC  ;--
                              09EC
                              09EC  UNIT_INIT:
         51    24 A5     D0   09EC          MOVL    UCB$L_CRB(R5),R1          ; GET POINTER TO CRB
                              09F0
                              09F0          ; DETERMINE IF THIS IS INITIAL LOADING OR POWER RECOVERY
      67 64 A5    05     E0   09F0          BBS     #UCB$V_POWER,UCB$W_STS(R5),60$  ; BRANCH IF POWER RECOVERY
                              09F5  ;
                              09F5  ;       D R I V E R   L O A D
                              09F5  ;
                              09F5          ; INITIALIZE INPUT QUEUE
   00AC C5    00AC C5    DE   09F5          MOVAL   UCB$L_INQFL(R5),UCB$L_INQFL(R5)
   00B0 C5    00AC C5    DE   09FC          MOVAL   UCB$L_INQFL(R5),UCB$L_INQBL(R5)
                              0A03
                              0A03          ; MAKE UCB OWNER OF IDB
         50    2C A1     D0   0A03          MOVL    CRB$L_INTD+VEC$L_IDB(R1),R0  ; GET POINTER TO IDB
         04 A0    55     D0   0A07          MOVL    R5,IDB$L_OWNER(R0)        ; MAKE UCB OWNER OF IDB
                              0A0B
                              0A0B          ; OPTIONALLY PREALLOCATE MAP REGISTERS
   53    00000000'GF     9A   0A0B          MOVZBL  G^IOC$GW_LAMAPREG,R3      ; NUM. TO PREALLOCATE (SYSGEN PARAM.)
                  30     13   0A12          BEQL    20$                       ; DON'T PREALLOCATE
      00FE 8F     53     B1   0A14          CMPW    R3,#254                   ; Prevent allocating more than 254.
                  05     15   0A19          BLEQ    10$                       ; LEQ implies we are OK.
   53    00FE 8F         3C   0A1B          MOVZWL  #254,R3                   ; Else reduce request to 254 registers.
                              0A20  10$:
     00000000'GF         16   0A20          JSB     G^IOC$ALOUBMAPRMN         ; Permanently allocate specified number.
               32 50     E9   0A26          BLBC    R0,50$                    ; ERROR - DIDN'T ALLOCATE
         51    24 A5     D0   0A29          MOVL    UCB$L_CRB(R5),R1          ; Refresh R1 => CRB.
             8000 8F     AA   0A2D          BICW    #VEC$M_MAPLOCK,-          ; Undo permanent bit set by IOC$ALOUBMAPRMN.
               34 A1          0A31                  CRB$L_INTD+VEC$W_MAPREG(R1)
             34 A1   D0  0A33          MOVL    CRB$L_INTD+VEC$W_MAPREG(R1),-   ; SAVE INFO. ON MAP REGISTERS
           00AB C5       0A36                  UCB$L_PREALLOC(R5)              ; ALLOCATED
                         0A39
                         0A39          ; NOW MARK IN UCB BITMAP AS AVAILABLE, THE MAP REGISTERS ALLOCATED
             50   00 D2  0A39          MCOML   #0,R0                           ; BITMAP PATTERN (1 MEANS AVAILABLE)
      54   00AB C5   3C  0A3C          MOVZWL  UCB$L_PREALLOC(R5),R4           ; R4 contains starting map register #
              FC01   30  0A41          BSBW    ALT_LOCALBITMAP                 ; ALTER MAP
                         0A44
                         0A44 20$:     ; ALLOCATE AND LOAD MAP REGISTERS TO PERMANENTLY MAP RDA IN UCB
                51   10  0A44          BSBB    LOADUCB                         ; LOAD BOFF, BCNT, AND SVAPTE IN UCB
              FB69   30  0A46          BSBW    SETMAPREG                       ; REQUEST AND LOAD UBA MAP REGISTERS
             0F 50   E9  0A49          BLBC    R0,50$                          ; ALLOCATION FAILURE
             34 A1   D0  0A4C          MOVL    CRB$L_INTD+VEC$W_MAPREG(R1),-   ; SAVE ALLOCATED MAP REGISTER
           00A4 C5       0A4F                  UCB$L_RDAMR(R5)                 ; INFO. IN UCB
      00A0 C5   52   D0  0A52          MOVL    R2,UCB$L_RDABA(R5)              ; UNIBUS ADDRESS OF RDA
        64 A5   10   A8  0A57          BISW    #UCB$M_ONLINE,UCB$W_STS(R5)     ; SET UNIT ONLINE
                     05  0A5B 50$:     RSB
                         0A5C
                         0A5C
                         0A5C      ;
                         0A5C      ;           P O W E R   R E C O V E R Y
                         0A5C      ;
        44 A5   01   CA  0A5C 60$:     BICL    #LA$M_MCVALID,UCB$L_DEVDEPEND(R5)  ; CLEAR MICROCODE VALID
        64 A5   10   A8  0A60          BISW    #UCB$M_ONLINE,UCB$W_STS(R5)     ; SET UNIT ONLINE
                         0A64
                         0A64          ; RELOAD UBA MAP REGISTERS TO MAP RDA IN UCB
                31   10  0A64          BSBB    LOADUCB                         ; LOAD BCNT, BOFF, AND SVAPTE IN UCB
           00A4 C5   D0  0A66          MOVL    UCB$L_RDAMR(R5),-               ; LOAD MAPREG, NUMREG, AND DATAPATH
             34 A1       0A6A                  CRB$L_INTD+VEC$W_MAPREG(R1)     ; IN CRB
       00000000'GF   16  0A6C          JSB     G^IOC$LOADUBAMAP                ; LOAD MAP REGISTERS
                         0A72
                         0A72          ; FORK TO COMPLETE ALL ACTIVE REQUESTS
           00CC C5   D5  0A72          TSTL    UCB$L_FORKP(R5)                 ; INTERLOCK AGAINST MULTIPLE PWR FAILS
                1E   12  0A76          BNEQ    90$                             ; IT'S ALREADY QUEUED!
      55   00CC C5   DE  0A78          MOVAL   UCB$L_FORKP(R5),R5              ; POINT TO FORK BLOCK
                         0A7D          FORK
      55   FF34 C5   DE  0A83          MOVAL   -UCB$L_FORKP(R5),R5             ; RESTORE POINTER TO UCB
           00CC C5   D4  0A88          CLRL    UCB$L_FORKP(R5)                 ; INDICATE THAT FORK BLOCK IS AVAILABLE
      50   0364 8F   3C  0A8C          MOVZWL  #SS$_POWERFAIL,R0               ; RETURN STATUS
                51   D4  0A91          CLRL    R1
              FF3F   30  0A93          BSBW    COMPLETE_ALL                    ; COMPLETE ALL REQUESTS
                     05  0A96 90$:     RSB
                         0A97
                         0A97
                         0A97
                         0A97          ; LOCAL SUBROUTINE TO LOAD BCNT, BOFF, AND SVAPTE FIELDS IN
                         0A97          ; UCB WITH VALUES WHICH DESCRIBE UCB$W_RDA
                         0A97
                         0A97 LOADUCB:
        7E A5   3A   B0  0A97          MOVW    #58,UCB$W_BCNT(R5)              ; SIZE OF RDA
      50   0164 C5   3E  0A9B          MOVAW   UCB$W_RDA(R5),R0                ; GET ADDRESS OF RDA
                         0AA0
                         0AA0          ASSUME  VA$S_BYTE EQ 9
7C A5   50   FE00 8F AB  0AA0          BICW3   #^XFE00,R0,UCB$W_BOFF(R5)       ; INSERT BYTE OFFSET IN PAGE
   50   50   15   09 EF  0AA7          EXTZV   #VA$V_VPN,#VA$S_VPN,R0,R0       ; GET VIRTUAL PAGE #
      52  00000000'GF D0 0AAC          MOVL    G^MMG$GL_SPTBASE,R2             ; GET ADDRESS OF SYSTEM PAGE TABLE
        78 A5   6240 DE  0AB3          MOVAL   (R2)[R0],UCB$L_SVAPTE(R5)       ; STORE SVA OF PTE FOR RDA
```

```
 05  0AB8  2344          RSB
```

```
                                     0AB9  2346
                                     0AB9  2347
                                     0AB9  2348 LA_END:                                 ; ADDRESS OF LAST LOCATION IN DRIVER
                                     0AB9  2349
                                     0AB9  2350
                                     0AB9  2351
                                     0AB9  2352
                                     0AB9  2353         .END
```

```
$$$                  = 00000020 R       02    EXE$FINISHIOC          ********     X   03
$$OP                 = 00000002               EXE$FORK               ********     X   03
ABORT                  000002CE R       03    EXE$GL_TENUSEC         ********     X   03
ACB$B_RMOD           = 0000000B               EXE$GL_UBDELAY         ********     X   03
ACB$L_AST            = 00000010               EXE$INSERTIRP          ********     X   03
ACB$L_ASTPRM         = 00000014               EXE$IOFORK             ********     X   03
ACB$L_PID            = 0000000C               EXE$QIORETURN          ********     X   03
ACB$M_QUOTA          = 00000040               EXE$READLOCKR          ********     X   03
ALIGNERR               000002C0 R       03    EXE$WRITECHK           ********     X   03
ALLOC_LOCALMR          000005E7 R       03    EXE$WRITELOCK          ********     X   03
ALT_LOCALBITMAP        00000645 R       03    EXE$WRITELOCKR         ********     X   03
AT$_UBA              = 00000001               FKB$B_FIPL           = 0000000B
BFRFULL                000007F5 R       03    FKB$K_LENGTH         = 00000018
CANCELCK               000009CA R       03    FUNCTABLE              00000038 R       03
CANCEL_IO              00000965 R       03    FUNCTAB_LEN          = 00000058
CLEANUP                000002E5 R       03    IDB$L_CSR            = 00000000
CMDERR                 000007B1 R       03    IDB$L_OWNER          = 00000004
COM$POST               ********     X   03    INITIALIZE             000003CB R       03
COMMON                 000003DE R       03    INIT_FDT               0000016E R       03
COMPLETE_ALL           000009D5 R       03    INTEXIT                000006DE R       03
COMPL_ALL_REQS         0000076D R       03    IO$V_SETEVF          = 00000006
CRB$L_INTD           = 00000024               IO$_INITIALIZE       = 00000004
CRB$L_INTD2          = 00000048               IO$_LOADMCODE        = 00000001
DC$_REALTIME         = 00000060               IO$_QSTOP            = 00000007
DDB$L_DDT            = 0000000C               IO$_SETCLOCK         = 00000037
DEV$M_AVL            = 00040000               IO$_SETCLOCKP        = 00000005
DEV$M_ELG            = 00400000               IO$_STARTDATA        = 00000038
DEV$M_IDV            = 04000000               IO$_STARTDATAP       = 00000006
DEV$M_ODV            = 08000000               IO$_STARTMPROC       = 00000002
DEV$M_RTM            = 20000000               IO$_STOP             = 00000003
DEV$M_SHR            = 00010000               IO$_VIRTUAL          = 0000003F
DEVADDR              = 00000002               IOC$ALOUBAMAP          ********     X   03
DODIAGERL              000008E7 R       03    IOC$ALOUBMAPRMN        ********     X   03
DONE                   0000042C R       03    IOC$DIAGBUFILL         ********     X   03
DPT$C_LENGTH         = 00000038               IOC$GW_LAMAPREG        ********     X   03
DPT$C_VERSION        = 00000004               IOC$INITIATE           ********     X   03
DPT$INITAB             00000038 R       02    IOC$LOADUBAMAP         ********     X   03
DPT$M_NOUNLOAD       = 00000004               IOC$MNTVER             ********     X   03
DPT$REINITAB           0000005D R       02    IOC$PURGDATAP          ********     X   03
DPT$TAB                00000000 R       02    IOC$RELDATAP           ********     X   03
DT$_LPA11            = 00000001               IOC$RELMAPREG          ********     X   03
DYN$C_ACB            = 00000002               IOC$REQDATAPNW         ********     X   03
DYN$C_CRB            = 00000005               IOC$RETURN             ********     X   03
DYN$C_DDB            = 00000006               IOC$WFIKPCH            ********     X   03
DYN$C_DPT            = 0000001E               IOFCTBL                00000090 R       03
DYN$C_FRK            = 00000008               IOFCTBLN             = 00000007
DYN$C_UCB            = 00000010               IPL$_QUEUEAST        = 00000006
EMB$B_DV_ERTCNT      = 00000010               IRP$B_CARCON         = 0000003C
EMB$L_DV_REGSAV      = 0000004E               IRP$B_EFN            = 00000022
EMB$Q_DV_IOSB        = 00000012               IRP$B_RMOD           = 0000000B
EMB$W_DV_STS         = 0000001A               IRP$B_TYPE           = 0000000A
ERL$DEVICERR           ********     X   03    IRP$C_LENGTH         = 000000C4
ERL$DEVICTMO           ********     X   03    IRP$L_ASTPRM         = 00000014
ERL$RELEASEMB          ********     X   03    IRP$L_BFR_AST        = 0000003C
ERROR                  00000748 R       03    IRP$L_IOQBL          = 00000004
EXE$ALONONPAGED        ********     X   03    IRP$L_IOQFL          = 00000000
EXE$DEANONPAGED        ********     X   03    IRP$L_IOST1          = 00000038
```

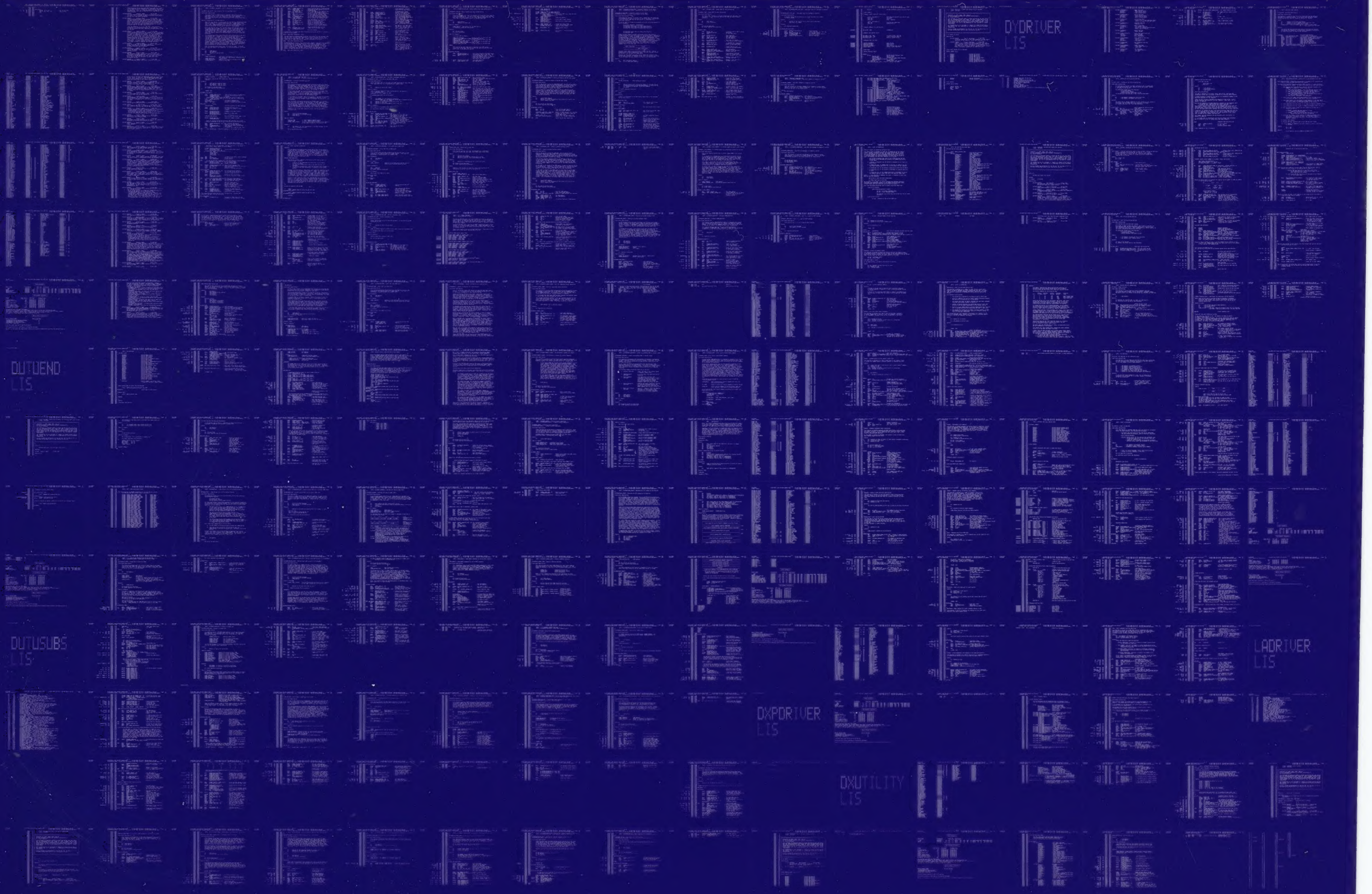
```
IRP$L_IOST2      = 0000003C             PRI$_IOCOM       = 00000001
IRP$L_MEDIA      = 00000038             QSTOP_FDT          000002F5 R      03
IRP$L_OVR_AST    = 00000040             QUEUE_STOP_REQ     00000810 R      03
IRP$L_PID        = 0000000C             QUE_PKT            0000031A R      03
IRP$L_RDAMAPREG  = 00000040             RDA_IN_UCB         000003C4 R      03
IRP$L_SEGVBN     = 00000048             READLOCK           000002D4 R      03
IRP$L_SIP        = 00000048             REL_MRDP           00000665 R      03
IRP$L_SVAPTE     = 0000002C             REQERR             0000078E R      03
IRP$S_FCODE      = 00000006             REQ_COMPLETE       0000050A R      03
IRP$W_ABCNT      = 00000040             RESET              00000130 R      03
IRP$W_CHAN       = 00000028             SCH$GL_PCBVEC      ********     X  03
IRP$W_FUNC       = 00000020             SCH$POSTEF         ********     X  03
IRP$W_SIZE       = 00000008             SCH$QAST           ********     X  03
LA$DDT             00000000 RG     03   SDATA              0000047A R      03
LA$M_MCVALID     = 00000001             SETCHAR            00000445 R      03
LA$RDYININTSV      000006B5 RG     03   SETCLOCK_FDT       000001BE R      03
LA$RDYOUTINTSV     000006E8 RG     03   SETMAPREG          000005B2 R      03
LA$S_CONFIG      = 0000000A             SET_CLOCK          000003AE R      03
LA$S_MCTYPE      = 00000002             SIGNAL_BFR_FULL    0000083F R      03
LA$S_RATE        = 00000003             SIP$B_BFR_DATAP    00000033
LA$V_CONFIG      = 00000003             SIP$B_BFR_NUMRE    00000032
LA$V_MCTYPE      = 00000001             SIP$B_RCL_DATAP    0000003F
LA$V_PRESET      = 00000010             SIP$B_RCL_NUMRE    0000003E
LA$V_RATE        = 0000000D             SIP$B_TYPE         0000000A
LA_CISR            00000000             SIP$B_USW_DATAP    00000027
LA_CISR_M_CRAM   = 00002000             SIP$B_USW_NUMRE    00000026
LA_CISR_M_ENA    = 00000800             SIP$B_VBFRMASK     00000007
LA_CISR_M_IE     = 00000040             SIP$L_BFR_SVAPT    00000028
LA_CISR_M_RESET  = 00004000             SIP$L_RCL_SVAPT    00000034
LA_CISR_M_ROMO   = 00000400             SIP$L_SLVDATA      0000000C
LA_CISR_M_RUN    = 00008000             SIP$L_USW_SVAPT    0000001C
LA_COSR            00000002             SIP$W_BCNT         00000002
LA_COSR_M_IE     = 00000040             SIP$W_BFR_BCNT     0000002E
LA_COSR_M_RDY    = 00000080             SIP$W_BFR_BOFF     0000002C
LA_COSR_S_ERRTP  = 00000002             SIP$W_BFR_MAPRE    00000030
LA_COSR_V_ERRTP  = 0000000D             SIP$W_MODE         00000000
LA_END             00000AB9 R      03   SIP$W_RCL_BCNT     0000003A
LA_MAINT           00000006             SIP$W_RCL_BOFF     00000038
LA_RDA             00000004             SIP$W_RCL_MAPRE    0000003C
LA_REGDUMP         00000950 R      03   SIP$W_SIZE         00000008
LENGTHERR          000002C7 R      03   SIP$W_USW_BCNT     00000022
LOAD               00000400 R      03   SIP$W_USW_BOFF     00000020
LOADUCB            00000A97 R      03   SIP$W_USW_MAPRE    00000024
LOAD_MICROCODE     00000097 R      03   SIZ...           = 00000001
MASKR            = 00000000             SS$_ABORT        = 0000002C
MASKL            = 00000080             SS$_BADPARAM     = 00000014
MCNVALID           000003F6 R      03   SS$_BUFNOTALIGN  = 00000324
MMG$GL_SPTBASE     ********     X  03   SS$_CANCEL       = 00000830
MMG$UNLOCK         ********     X  03   SS$_CTRLERR      = 00000054
NO_ERROR           000007C1 R      03   SS$_DATACHECK    = 0000005C
P1               = 00000000             SS$_DEVACTIVE    = 000002C4
P2               = 00000004             SS$_DEVCMDERR    = 0000032C
P3               = 00000008             SS$_DEVREQERR    = 00000334
P4               = 0000000C             SS$_EXQUOTA      = 0000001C
PCB$L_PID        = 00000060             SS$_INSFBUFDP    = 0000033C
PCB$W_ASTCNT     = 00000038             SS$_INSFMAPREG   = 00000344
PR$_IPL          = 00000012             SS$_INSFMEM      = 00000124
```

```
SS$_IVBUFLEN                  = 0000034C               UCB$W_STS                 = 00000064
SS$_IVMODE                    = 00000354               UNIT_INIT                   000009EC R      03
SS$_MCNOTVALID                = 0000035C               UNLOCK                      0000057C R      03
SS$_NORMAL                    = 00000001               UNLOCKF                     00000572 R      03
SS$_PARITY                    = 000001F4               VA$S_BYTE                 = 00000009
SS$_POWERFAIL                 = 00000364               VA$S_VPN                  = 00000015
SS$_TIMEOUT                   = 0000022C               VA$V_VPN                  = 00000009
STARTDATA_FDT                   000001D6 R      03     VEC$B_DATAPATH            = 00000013
STARTIO                         00000342 R      03     VEC$B_NUMREG              = 00000012
STARTMP_FDT                     00000165 R      03     VEC$L_IDB                 = 00000008
STARTREQ                        000007C3 R      03     VEC$L_UNITINIT            = 00000018
START_DATA                      000003B6 R      03     VEC$M_LWAE                = 00000020
STOP                            000003BE R      03     VEC$M_MAPLOCK             = 00008000
STOP_MODE                     = 00000003               VEC$S_DATAPATH            = 00000005
STRT_NXT_REQ                    00000431 R      03     VEC$V_DATAPATH            = 00000000
TIMEOUT                         00000758 R      03     VEC$W_MAPREG              = 00000010
UCB$B_DEVCLASS                = 00000040               WAIT                        00000407 R      03
UCB$B_DEVTYPE                 = 00000041               WRITELOCK                   000002DC R      03
UCB$B_DIPL                    = 0000005E
UCB$B_ERTCNT                  = 00000080
UCB$B_FIPL                    = 0000000B
UCB$K_SIZE                    = 000001A0
UCB$L_CRB                     = 00000024
UCB$L_DEVCHAR                 = 00000038
UCB$L_DEVDEPEND               = 00000044
UCB$L_DPC                     = 0000009C
UCB$L_DUETIM                  = 0000006C
UCB$L_EMB                     = 00000094
UCB$L_FORKO                     000000B4
UCB$L_FORKP                     000000CC
UCB$L_FPC                     = 0000000C
UCB$L_FR3                     = 00000010
UCB$L_INQBL                     000000B0
UCB$L_INQFL                     000000AC
UCB$L_IRP                     = 00000058
UCB$L_PREALLOC                  000000A8
UCB$L_RDABA                     000000A0
UCB$L_RDAMR                     000000A4
UCB$L_REGSAVE                   000000E4
UCB$L_RQLIST                    00000104
UCB$L_SVAPTE                  = 00000078
UCB$M_BSY                     = 00000100
UCB$M_CANCEL                  = 00000008
UCB$M_ONLINE                  = 00000010
UCB$M_POWER                   = 00000020
UCB$V_BSY                     = 00000008
UCB$V_CANCEL                  = 00000003
UCB$V_ERLOGIP                 = 00000002
UCB$V_INT                     = 00000001
UCB$V_POWER                   = 00000005
UCB$W_BCNT                    = 0000007E
UCB$W_BOFF                    = 0000007C
UCB$W_FUNC                    = 0000009A
UCB$W_MRBITMAP                  00000124
UCB$W_RDA                       00000164
UCB$W_RISAVE                    000000F4
UCB$W_ROSAVE                    000000FC
```

```
                                           +----------------+
                                           ! Psect synopsis !
                                           +----------------+

PSECT name                       Allocation          PSECT No.  Attributes
----------                       ----------          ---------  ----------
.  ABS  .                        00000000  (     0.)  00 (   0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                            000001A0  (   416.)  01 (   1.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$$$105_PROLOGUE                  00000072  (   114.)  02 (   2.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$$$115_DRIVER                    00000AB9  (  2745.)  03 (   3.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC LONG

                                        +------------------------+
                                        ! Performance indicators !
                                        +------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     30    00:00:00.07    00:00:01.07
Command processing                108    00:00:00.40    00:00:03.44
Pass 1                            635    00:00:19.53    00:01:10.65
Symbol table sort                   0    00:00:02.70    00:00:11.54
Pass 2                            388    00:00:04.96    00:00:16.90
Symbol table output                17    00:00:00.19    00:00:01.19
Psect synopsis output               0    00:00:00.00    00:00:00.02
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals             1180    00:00:27.86    00:01:44.81

The working set limit was 2250 pages.
166852 bytes (326 pages) of virtual memory were used to buffer the intermediate code.
There were 130 pages of symbol table space allocated to hold 2487 non-local and 98 local symbols.
2353 source lines were read in Pass 1, producing 23 object records in Pass 2.
51 pages of virtual memory were used to define 48 macros.

                                       +--------------------------+
                                       ! Macro library statistics !
                                       +--------------------------+

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                  34
_$255$DUA28:[SYSLIB]STARLET.MLB;2               11
TOTALS (all libraries)                          45

2717 GETS were required to define 45 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:LADRIVER/OBJ=OBJ$:LADRIVER MSRC$:LADRIVER/UPDATE=(ENH$:LADRIVER)+EXECML$/LIB
```

0111 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0112 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY